Santiago Dynamical System for Convergence Analysis of Automated Test-Case Reduction

Mendoza Delgado Raphael Santiago, Est.[1]; Rojas Orbegoso Jorge Luis, Mg.[1]; Ruiz Salazar Jenny María, Dra.[2]; Arbañil Rivadeneira Rubén Orlando, Mg.[1]; Delgado Baltazar Marisol Paola, Mg.[1]; Farfán Aguilar José Antonio, Dr.[1]; Garrido Huaylinos Luisa Rosario, Dra.[3]; [1]Universidad Nacional del Callao, Perú, rsmendozad@unac.edu.pe, jlrojaso@unac.edu.pe, roarbanilr@unac.edu.pe, mpdelgadob@unac.edu.pe, jafarfana@unac.edu.pe, [2]Universidad Nacional Federico Villareal, Perú, jruizs@unfv.edu.pe, [3]Universidad Nacional de Educación Enrique Guzmán y Valle, Perú, lgarrido@une.edu.pe.

Abstract—This paper introduces a Santiago Dynamical System (SDS) for analyzing the convergence of automated test-case reduction. The state is a failure-inducing test case, the transition is a reduction operator that preserves the failure predicate, and length is used as a numerical observable. The study follows a quantitative, repeated-measures computational design and compares three idealized reduction strategies representing generic Delta Debugging, hierarchical reduction, and syntax-guided reduction. A reproducible synthetic benchmark of 90 failure-inducing cases, ranging from 250 to 15,000 tokens, was generated to evaluate final size, stabilization iterations, oracle queries, execution time, reduction percentage, and average contraction ratio. Shapiro–Wilk tests showed non-normality in the principal outcomes; consequently, Friedman tests and Holm-adjusted paired Wilcoxon tests were applied. Significant differences were obtained in final size ($\chi^2 = 103.02$), stabilization iterations ($\chi^2 = 174.63$), oracle queries ($\chi^2 = 161.62$), and execution time ($\chi^2 = 138.47$), with $p < 0.001$ in every case. The syntax-guided strategy resulted in the smallest final median size and the most efficient convergence. The proposed framework connects test-case reduction with functional graphs, fixed points, basins of attraction, and discrete Lyapunov arguments, providing a mathematical basis for comparing reducers and designing convergence-aware software-engineering tools. Because the empirical data are simulated, the results constitute a proof of concept and require validation on public and industrial fault corpora.

Keywords: test-case reduction, delta debugging, discrete dynamical systems, software testing, convergence analysis.

Sistema Dinámico de Santiago para Analizar la Convergencia de la Reducción Automática de Casos de Prueba

Resumen—Se propone un Sistema Dinámico de Santiago (SDS) para analizar la convergencia de la reducción automática de casos de prueba. El estado es un caso que reproduce una falla, la transición es un operador de reducción que preserva el predicado de fallo y la longitud actúa como observable numérica. Se desarrolló un estudio cuantitativo computacional de medidas repetidas que comparó tres estrategias idealizadas: reducción genérica tipo Delta Debugging, reducción jerárquica y reducción guiada por sintaxis. Se generó un banco sintético reproducible de 90 casos, con longitudes iniciales entre 250 y 15,000 tokens. Las variables fueron tamaño final, iteraciones de estabilización, consultas al oráculo, tiempo, porcentaje de reducción y razón media de contracción. Las pruebas de Shapiro–Wilk evidenciaron ausencia de normalidad en los resultados principales; por ello, se aplicaron las pruebas de Friedman y Wilcoxon pareada con corrección de Holm. Se encontraron diferencias significativas en tamaño final ($\chi^2 = 103{,}02$), iteraciones ($\chi^2 = 174{,}63$), consultas ($\chi^2 = 161{,}62$) y tiempo ($\chi^2 = 138{,}47$), con $p < 0{,}001$. La estrategia guiada por sintaxis presentó el menor tamaño final mediano y la convergencia más eficiente. Los resultados son una prueba de concepto basada en simulación y deberán validarse con corpus públicos e industriales.

Palabras clave: reducción de casos de prueba, depuración delta, sistemas dinámicos discretos, pruebas de software, análisis de convergencia.

I. Introducción

I-A. Realidad problemática

Los defectos de software suelen manifestarse mediante entradas extensas, secuencias de interacción, programas fuente, documentos estructurados o configuraciones cuya mayor parte no interviene causalmente en la falla. Cuando el caso que reproduce el defecto contiene miles de tokens, líneas o nodos sintácticos, la localización del elemento responsable exige múltiples ejecuciones, inspecciones manuales y ciclos de comunicación entre equipos de desarrollo y aseguramiento de calidad. La reducción automática busca transformar una entrada grande en otra considerablemente menor que conserve el mismo comportamiento de interés, facilitando la reproducción, el diagnóstico y la reparación del defecto [1], [3].

Delta Debugging sistematizó la búsqueda de circunstancias inductoras de fallos mediante divisiones y pruebas sucesivas, hasta alcanzar una entrada 1-mínima respecto del conjunto de transformaciones permitido [1]. Sin embargo, la reducción puramente textual puede generar numerosos candidatos inválidos o ignorar la estructura jerárquica de la entrada. Para superar estas limitaciones se propusieron enfoques basados en árboles, como Hierarchical Delta Debugging (HDD), reductores especializados como C-Reduce y métodos guiados por gramática como Perses [2], [3], [4]. Estudios posteriores han extendido la reducción jerárquica con operaciones de sustitución, elevación de subárboles y reconstrucción de dependencias, mostrando que el problema sigue activo y que existe una tensión entre tamaño final, costo del oráculo, tiempo y generalidad del reductor [6], [7], [8].

A pesar de estos avances, la literatura suele comparar reductores mediante indicadores aislados: tamaño del resultado, tiempo total o número de ejecuciones. Falta un marco unificador que describa la reducción como una trayectoria, identifique su punto de estabilización y relacione las tasas de contracción con la estructura del caso. En este trabajo se plantea que la reducción puede estudiarse como un sistema dinámico discreto: cada iteración produce un nuevo caso, la secuencia forma una órbita y el caso irreducible alcanzado es un punto fijo del operador empleado. Esta interpretación permite trasladar conceptos de grafos funcionales, atractores, cuencas y funciones de Lyapunov al análisis de herramientas de Ingeniería de Software [9], [10], [11].

En este trabajo, la denominación Sistema Dinámico de Santiago (SDS) corresponde a una formulación aplicada que integra conceptos establecidos de sistemas dinámicos discretos al análisis de la reducción automática de casos de prueba; no pretende introducir una nueva teoría de sistemas dinámicos.

La contribución principal no consiste en proponer un nuevo algoritmo de reducción, sino en integrar trayectoria, punto fijo, velocidad de convergencia y costo computacional dentro de un marco común para analizar y comparar procesos de reducción.

I-B. Marco teórico

Sea $\mathcal{T}$ el espacio de casos de prueba y sea $P : \mathcal{T} \to \{0, 1\}$ un oráculo tal que $P(x) = 1$ cuando x reproduce la falla. El subconjunto relevante es

$$\mathcal{T}_P = \{x \in \mathcal{T} : P(x) = 1\}. \tag{1}$$

Un reductor determinista es una aplicación $R : \mathcal{T}_P \to \mathcal{T}_P$ que preserva el predicado. A partir de $x_0 \in \mathcal{T}_P$ se define la órbita

$$x_{k+1} = R(x_k), \qquad k \geq 0. \tag{2}$$

La longitud $\ell : \mathcal{T}_P \to \mathbb{N}$ puede medir bytes, caracteres, tokens, líneas o nodos. La secuencia observable es

$$n_k = \ell(x_k), \qquad n_{k+1} = \ell(R(x_k)). \tag{3}$$

El sistema $(\mathcal{T}_P, R)$ es el objeto dinámico completo; (n_k) es su proyección de longitud. Esta distinción es necesaria porque dos casos con la misma longitud pueden reducirse de manera diferente. Solo cuando la evolución depende exclusivamente del tamaño existe una función escalar bien definida $S : \mathbb{N} \to \mathbb{N}$ con $n_{k+1} = S(n_k)$. En el modelo experimental se emplea un sustituto escalar controlado para comparar perfiles de

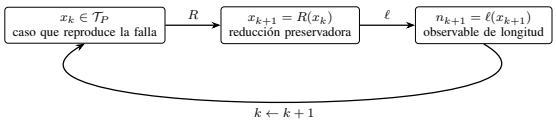


Figura 1. Representación del SDS de reducción automática y su observable de longitud.

convergencia, sin afirmar que la longitud capture toda la semántica del caso.

La reducción genérica tipo DD no requiere una gramática, pero puede evaluar candidatos sintácticamente inválidos [1]. HDD aprovecha una representación arbórea para eliminar subestructuras por niveles [2], [5]. C-Reduce combina numerosas transformaciones especializadas para programas C/C++ [3]. Perses restringe la búsqueda a programas sintácticamente válidos usando gramáticas formales y ha mostrado mejoras en calidad y eficiencia frente a reductores jerárquicos generales [4]. Estas diferencias justifican representar cada estrategia por un operador R_j y comparar sus órbitas sobre los mismos estados iniciales.

I-C. Definición conceptual de las variables

La variable independiente es la *estrategia de reducción*, con tres niveles: DD genérico, HDD estructural y reducción guiada por sintaxis. La variable dependiente es la *eficiencia de convergencia del SDS*, entendida como la capacidad de alcanzar un punto fijo pequeño con pocas iteraciones, consultas al oráculo y tiempo. Se operacionaliza mediante: tamaño final N_f; tiempo de estabilización τ; número de consultas Q; tiempo computacional T; porcentaje de reducción $100(1 - N_f/N_0)$; y razón media de contracción $\bar{\rho} = \tau^{-1} \sum n_{k+1}/n_k$.

I-D. Planteamiento de hipótesis

La hipótesis general sostiene que la estrategia de reducción produce diferencias significativas en el perfil de convergencia del SDS. Las hipótesis específicas son:

- H_{1a}: existen diferencias significativas en el tamaño final alcanzado por DD, HDD y la reducción guiada por sintaxis.
- H_{1b}: existen diferencias significativas en el número de iteraciones necesarias para alcanzar un punto fijo.
- H_{1c}: existen diferencias significativas en las consultas al oráculo requeridas por las tres estrategias.
- H_{1d}: existen diferencias significativas en el tiempo computacional de reducción.

Las hipótesis nulas correspondientes establecen igualdad de distribuciones entre las tres condiciones. Se adoptó un nivel de significación $\alpha = 0{,}05$.

II. Materiales y Métodos

II-A. Enfoque, alcance y diseño

El estudio adopta un enfoque cuantitativo, alcance explicativo-comparativo y diseño computacional experimental de medidas repetidas. Cada caso inicial fue procesado bajo las tres estrategias, de modo que el propio caso actuó como bloque y se redujo la variabilidad atribuible al tamaño o dificultad inicial. El trabajo es una prueba de concepto metodológica: los datos se generaron mediante simulación reproducible y no representan mediciones de herramientas ejecutadas sobre un corpus industrial. Esta delimitación evita atribuir rendimiento empírico real a los reductores y permite examinar si los indicadores propuestos discriminan perfiles dinámicos plausibles.

II-B. Población, muestra y unidad de análisis

La población conceptual está formada por casos de prueba finitos que reproducen una propiedad observable y admiten transformaciones reductoras. Se trabajó con una muestra sintética de 90 casos. Las longitudes iniciales se generaron con una distribución lognormal truncada entre 250 y 15,000 tokens, apropiada para representar una mayoría de casos medianos y una cola de entradas extensas. La unidad de análisis fue la trayectoria algoritmo–caso, obteniéndose 270 órbitas observadas.

II-C. Modelo de simulación

Para cada caso i, algoritmo j e iteración k, la longitud se modeló mediante

$$n_{i,j,k+1} = \max\{m_{ij}, \lceil m_{ij} + c_j(n_{i,j,k} - m_{ij}) + \varepsilon_{i,j,k} \rceil\}, \tag{4}$$

con $0 < c_j < 1$, piso de estabilización m_{ij} y perturbación de media cero $\varepsilon_{i,j,k}$. Los valores centrales de contracción fueron $c_{DD} = 0{,}80$, $c_{HDD} = 0{,}66$ y $c_{SG} = 0{,}54$. Los pisos medianos fueron 82, 50 y 30 tokens, respectivamente. Se incorporó una probabilidad pequeña de iteraciones de baja ganancia para representar candidatos rechazados, dependencias y regiones de búsqueda difíciles. La semilla pseudoaleatoria fue 731902.

La simulación finalizó cuando la longitud alcanzó una vecindad del piso asignado o cuando se alcanzó el número máximo de iteraciones establecido. Las consultas al oráculo fueron generadas como una función creciente del número de iteraciones, del logaritmo del tamaño inicial y de un factor de dificultad compartido por caso. El tiempo se calculó a partir de las consultas y del tamaño inicial con ruido multiplicativo. Estos mecanismos producen asimetría y heterocedasticidad, rasgos comunes en métricas de rendimiento de software [12].

Los componentes aleatorios restantes se generaron reproduciblemente a partir de esta semilla, manteniéndose las condiciones compartidas por caso para las tres estrategias.

II-D. Instrumentos y procedimiento

Se empleó una matriz computacional de registro con identificador del caso, algoritmo, longitud inicial, longitud final, iteraciones, consultas, tiempo, porcentaje de reducción y razón media. El procedimiento comprendió: generación de los casos; asignación de parámetros compartidos de dificultad; ejecución de las tres trayectorias; verificación de monotonicidad no creciente; cálculo de indicadores; análisis estadístico; y representación gráfica. La implementación se realizó con NumPy y SciPy, bibliotecas ampliamente utilizadas en computación científica [13], [14].

Algorithm 1 Simulación de una trayectoria SDS para el reductor j.

Entrada: Tamaño inicial n_0, contracción c_j, piso m, máximo K
Salida: Trayectoria $(n_0, \ldots, n_\tau)$ y tiempo τ

```
k ← 0
while k < K y n_k > m do
    Generar ε_k y un posible paso de baja ganancia
    n_{k+1} ← máx{m, ⌈m + c_j(n_k − m) + ε_k⌉}
    if n_{k+1} ≥ n_k then
        Forzar una disminución mínima de una unidad
    end if
    k ← k + 1
end while
τ ← k
return (n_0, ..., n_τ), τ
```

II-E. Análisis estadístico

Se calcularon la mediana y el rango intercuartílico debido a la naturaleza asimétrica de las métricas analizadas. Considerando el diseño de medidas repetidas, la naturaleza discreta de algunas variables y la presencia de asimetría y valores extremos, se adoptó un enfoque no paramétrico para los contrastes inferenciales. La prueba de Shapiro–Wilk se utilizó como diagnóstico complementario de la distribución de los datos [15]. Como cada caso fue evaluado bajo tres condiciones, las diferencias globales se contrastaron con Friedman [16]; el tamaño del efecto se informó mediante W de Kendall. Las comparaciones por pares utilizaron Wilcoxon de rangos con signo y corrección secuencial de Holm [17], [18]. Se consideraron efectos grandes cuando $W \geq 0{,}50$ y se interpretaron los resultados junto con las diferencias sustantivas de mediana.

II-F. Consideraciones de validez

La validez interna se fortalece mediante el diseño pareado y la semilla fija. La validez de constructo se limita porque una trayectoria de longitud no captura sintaxis, semántica ni equivalencia causal completa. La validez externa es deliberadamente restringida: los resultados no deben extrapolarse a implementaciones reales sin replicar el estudio con corpus de fallos, oráculos y herramientas ejecutables. El modelo busca evaluar el marco SDS y formular predicciones verificables, no sustituir la experimentación empírica.

III. Resultados

III-A. Resultados teóricos de convergencia

Proposición 1 (terminación por descenso estricto). Si $R : \mathcal{T}_P \to \mathcal{T}_P$ satisface $R(x) \neq x \Rightarrow \ell(R(x)) < \ell(x)$, entonces toda órbita alcanza un punto fijo en un número finito de iteraciones.

Demostración. Mientras x_k no sea fijo, la sucesión $\ell(x_k)$ decrece estrictamente en $\mathbb{N}$. Por el principio del buen orden no existe una cadena infinita estrictamente decreciente de naturales. Por tanto, existe τ tal que $R(x_\tau) = x_\tau$. □

Cuadro I
Resultados descriptivos: mediana [RIQ], $n = 90$.

Indicador	DD	HDD	Guiado por sintaxis
Tamaño final	77.5 [56.0–103.0]	47.0 [38.2–64.5]	27.5 [23.0–36.0]
Iteraciones	26.0 [23.0–29.8]	17.0 [16.0–19.0]	13.0 [12.0–13.0]
Consultas	5102.5 [3350–8955]	2281 [1393–3175]	1045 [745–1760]
Tiempo (s)	602.8 [236.3–1069]	166.4 [76.3–289.8]	87.7 [44.7–176.3]
Reducción (%)	95.70 [92.93–97.82]	97.57 [95.89–98.68]	98.50 [97.51–99.18]
$\bar{\rho}$	0.886 [0.873–0.899]	0.812 [0.792–0.835]	0.730 [0.710–0.750]

Este resultado formaliza una propiedad implícita de los reductores que aceptan únicamente transformaciones preservadoras y estrictamente menores. El punto fijo es irreducible respecto del repertorio de transformaciones de R, pero no necesariamente mínimo global, distinción central en Delta Debugging y reducción especializada [1], [3].

Proposición 2 (cota afín). Si $n_{k+1} \leq cn_k + b$, con $0 < c < 1$, entonces

$$n_k \leq c^k n_0 + \frac{b(1 - c^k)}{1 - c}. \tag{5}$$

En particular,

$$\limsup_{k \to \infty} n_k \leq \frac{b}{1 - c}.$$

Demostración. La sustitución recursiva produce $n_k \leq c^k n_0 + b \sum_{r=0}^{k-1} c^r$; la suma geométrica conduce a (5). □

La cota muestra convergencia geométrica hacia una región de radio controlado por $b/(1 - c)$. En ingeniería, c resume la capacidad reductora por ronda y b representa restricciones estructurales o costos residuales.

III-B. Resultados descriptivos

El Cuadro I presenta las medianas y rangos intercuartílicos. Los tres reductores disminuyeron sustancialmente la longitud inicial; sin embargo, los perfiles fueron diferentes. La reducción guiada por sintaxis alcanzó un tamaño final mediano de 27.5 tokens, frente a 47.0 en HDD y 77.5 en DD. También requirió menos iteraciones y consultas. El porcentaje mediano de reducción fue 98.50 % en el modelo guiado por sintaxis, 97.57 % en HDD y 95.70 % en DD.

La Figura 2 muestra la trayectoria mediana. DD presenta una cola prolongada de pequeñas reducciones; HDD se estabiliza aproximadamente alrededor de la iteración 20; y el modelo guiado por sintaxis alcanza su región atractora cerca de la iteración 13. El comportamiento concuerda con la interpretación de c_j: un valor menor genera descenso más rápido y un piso reducido.

III-C. Análisis de normalidad

El Cuadro II resume Shapiro–Wilk. El tamaño final fue no normal en las tres condiciones. Las consultas y el tiempo también presentaron $p < 0{,}05$ en todos los grupos. En iteraciones, DD y HDD no rechazaron normalidad, pero la condición guiada por sintaxis sí lo hizo. Estos resultados fueron consistentes con el uso previamente establecido de procedimientos no paramétricos para el diseño de medidas repetidas.

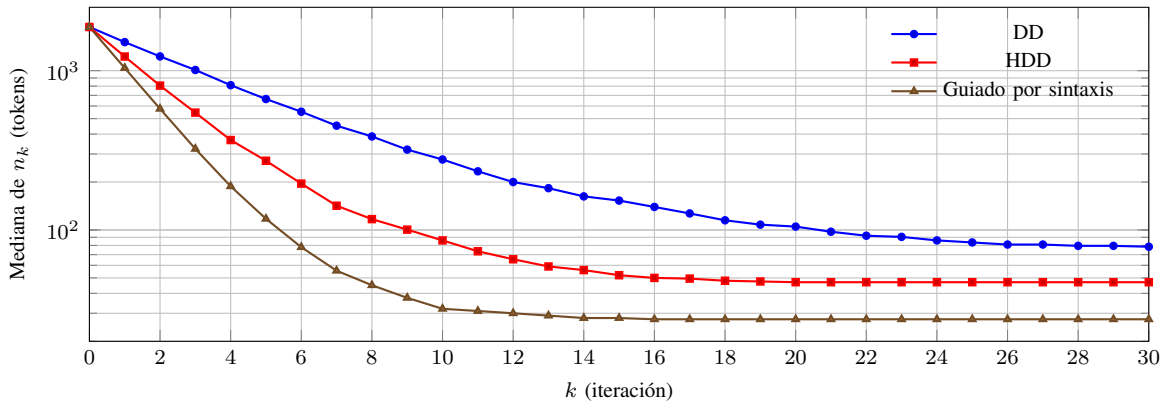


Figura 2. Trayectorias medianas simuladas de longitud para las tres estrategias de reducción.

Cuadro II
PRUEBA DE SHAPIRO–WILK POR ESTRATEGIA.

Variable	Estrategia	W	p
Tamaño final	DD	0.919	$< 0{,}001$
	HDD	0.875	$< 0{,}001$
	Guiada	0.951	0.002
Iteraciones	DD	0.980	0.188
	HDD	0.978	0.127
	Guiada	0.938	$< 0{,}001$
Consultas	DD	0.780	$< 0{,}001$
	HDD	0.783	$< 0{,}001$
	Guiada	0.790	$< 0{,}001$
Tiempo	DD	0.481	$< 0{,}001$
	HDD	0.287	$< 0{,}001$
	Guiada	0.482	$< 0{,}001$

Cuadro III
CONTRASTES GLOBALES DE FRIEDMAN.

Variable	$\chi^2(2)$	p	Kendall W
Tamaño final	103.02	$< 0{,}001$	0.572
Iteraciones	174.63	$< 0{,}001$	0.970
Consultas	161.62	$< 0{,}001$	0.898
Tiempo	138.47	$< 0{,}001$	0.769
Reducción porcentual	103.02	$< 0{,}001$	0.572
Razón media $\bar{\rho}$	176.09	$< 0{,}001$	0.978

III-D. Resultados inferenciales y contrastación

La prueba de Friedman rechazó las cuatro hipótesis nulas (Cuadro III). El tamaño del efecto fue grande en todos los indicadores, desde $W = 0{,}572$ para tamaño final hasta $W = 0{,}970$ para iteraciones. En consecuencia, la estrategia explica una proporción sustantiva del ordenamiento intra-caso de los resultados.

Las comparaciones de Wilcoxon con Holm confirmaron diferencias entre todos los pares en tamaño final, iteraciones, consultas y tiempo ($p_{Holm} < 0{,}001$). Para tamaño final, los efectos, expresados mediante la correlación de rango biserial (r_{rb}), fueron 0.741 entre DD y HDD, 0.997 entre DD y guiada, y 0.854 entre HDD y guiada. Los signos indican que DD dejó casos mayores y la estrategia guiada produjo los menores. En iteraciones, los efectos fueron prácticamente máximos: 1.000, 1.000 y 0.990, respectivamente.

Cuadro IV
COMPARACIONES DE WILCOXON CON CORRECCIÓN DE HOLM.

Variable	Comparación	p_{Holm}	r_{rb}
Tamaño final	DD–HDD	$< 0{,}001$	0.741
	DD–Guiada	$< 0{,}001$	0.997
	HDD–Guiada	$< 0{,}001$	0.854
Iteraciones	DD–HDD	$< 0{,}001$	1.000
	DD–Guiada	$< 0{,}001$	1.000
	HDD–Guiada	$< 0{,}001$	0.990
Consultas	DD–HDD	$< 0{,}001$	0.999
	DD–Guiada	$< 0{,}001$	1.000
	HDD–Guiada	$< 0{,}001$	0.911
Tiempo	DD–HDD	$< 0{,}001$	0.972
	DD–Guiada	$< 0{,}001$	1.000
	HDD–Guiada	$< 0{,}001$	0.681

Por tanto, se rechazaron las hipótesis nulas correspondientes y se encontró evidencia estadísticamente significativa a favor de H1a, H1b, H1c y H1d. Bajo los parámetros del modelo simulado, las estrategias presentaron diferencias en tamaño final, iteraciones de estabilización, consultas al oráculo y tiempo computacional. La estrategia guiada por sintaxis mostró las menores medianas, seguida de HDD y DD.

El Cuadro IV resume los contrastes pareados. Todos los valores ajustados permanecieron por debajo de 0.001. El patrón ordenado DD $>$ HDD $>$ guiada en tamaño, iteraciones, consultas y tiempo fue consistente en la gran mayoría de casos. Esta consistencia es relevante porque el contraste de Friedman identifica una diferencia global, pero no determina entre qué pares se produce; las pruebas posteriores permiten establecer la jerarquía de desempeño bajo el modelo simulado.

IV. APLICACIÓN EN INGENIERÍA DE SISTEMAS

IV-A. Arquitectura de una herramienta basada en SDS

La aplicación propuesta es una herramienta de reducción y análisis de convergencia que recibe un caso reproducible, un oráculo ejecutable y una colección de transformaciones. Su propósito no es únicamente producir una entrada menor,

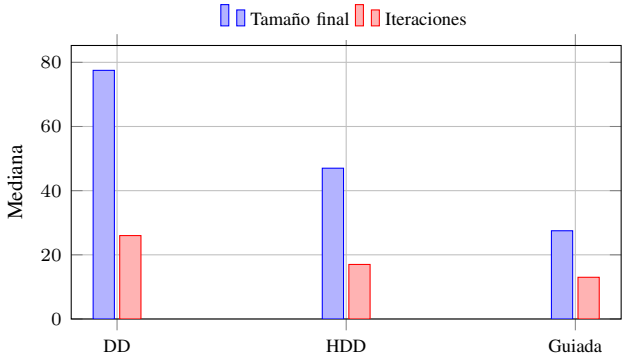


Figura 3. Comparación de medianas del tamaño final y las iteraciones de estabilización.

sino registrar la trayectoria completa, estimar la cercanía a la estabilización y comparar estrategias. La arquitectura comprende cinco componentes: repositorio de casos; ejecutor del oráculo; motor de reducción; monitor SDS; y gestor de resultados. El repositorio conserva la entrada original, metadatos, versión del sistema y configuración de ejecución. El oráculo determina si cada candidato mantiene la propiedad. El motor genera transformaciones textuales, jerárquicas o sintácticas. El monitor calcula observables y detecta puntos fijos. Finalmente, el gestor publica el caso reducido, la trayectoria y un reporte reproducible.

El monitor no debe suponer que toda disminución de longitud es válida. La transición solo se incorpora a la órbita cuando el oráculo confirma $P(x_{k+1}) = 1$. Por consiguiente, la secuencia almacenada es la de candidatos aceptados, mientras que las consultas rechazadas se contabilizan en Q. Esta separación permite distinguir la eficiencia dinámica de la eficiencia de búsqueda: un reductor puede presentar pocas iteraciones aceptadas, pero muchas evaluaciones fallidas.

IV-B. Escenarios de uso

La herramienta puede aplicarse a defectos de compiladores, analizadores estáticos, APIs, consultas de bases de datos, documentos XML/JSON, secuencias de interfaz y entradas de modelos de aprendizaje automático. En compiladores, el estado es un programa que provoca un fallo o una discrepancia de salida; en una API, es una solicitud que reproduce una excepción; en una base de datos, una consulta o conjunto de transacciones; y en una interfaz, una secuencia de acciones. La observable puede adaptarse: tokens para programas, nodos para documentos, operaciones para secuencias y elementos para conjuntos de datos. La versatilidad de Delta Debugging para diferentes clases de circunstancias y la especialización de C-Reduce y Perses muestran que la reducción es un patrón de ingeniería transversal [1], [3], [4].

Un caso de uso típico comienza con una incidencia cuyo archivo adjunto contiene miles de líneas. El ingeniero define un script que devuelve éxito solo si se reproduce la falla. La herramienta ejecuta varias estrategias y registra sus órbitas. Si una estrategia alcanza un punto fijo de 40 tokens después de 18 iteraciones y 2,000 consultas, mientras otra obtiene 30 tokens con 20,000 consultas, el equipo puede preferir la primera cuando el oráculo es costoso. Por ello, el tamaño final no debe ser el único criterio de selección.

IV-C. Indicadores de decisión y criterio multiobjetivo

Se propone la eficiencia de reducción por consulta

$$E_Q = \frac{N_0 - N_f}{Q}, \tag{6}$$

y la eficiencia temporal

$$E_T = \frac{N_0 - N_f}{T}. \tag{7}$$

Para comparar estrategias con prioridades distintas puede emplearse un costo normalizado

$$J_j = w_1 \frac{N_{f,j}}{N_0} + w_2 \frac{\tau_j}{\tau_{\text{máx}}} + w_3 \frac{Q_j}{Q_{\text{máx}}} + w_4 \frac{T_j}{T_{\text{máx}}}, \tag{8}$$

con $w_r \geq 0$ y $\sum w_r = 1$. La mejor estrategia minimiza J_j. En un entorno de investigación puede asignarse mayor peso a N_f; en mantenimiento industrial, donde cada ejecución tarda varios minutos, el peso dominante puede corresponder a Q o T. La formulación hace explícitos los compromisos y evita declarar ganador a un algoritmo basándose en una única métrica.

También se propone una regla de detención temprana. Sea $g_k = 1 - n_{k+1}/n_k$ la ganancia relativa. Si $g_k < \delta$ durante r aceptaciones consecutivas y el costo esperado de nuevas consultas excede el beneficio estimado, el proceso puede detenerse aunque todavía no haya probado irreducibilidad completa. Esta política produce un punto casi fijo y resulta útil cuando el objetivo es obtener rápidamente un caso comprensible. La herramienta debe reportar que se aplicó detención aproximada para no confundir el resultado con una salida 1-mínima.

IV-D. Requisitos de implementación

El Cuadro V resume requisitos funcionales y de calidad. La reproducibilidad exige almacenar la versión del sistema bajo prueba, el oráculo, las transformaciones y la semilla cuando exista aleatoriedad. El aislamiento es esencial porque los candidatos pueden provocar bloqueos, consumo excesivo o efectos externos. Los procesos deben ejecutarse con límites de tiempo, memoria y permisos. Asimismo, un mecanismo de caché evita repetir consultas sobre candidatos equivalentes, y la paralelización puede disminuir tiempo sin alterar la definición de la órbita aceptada.

IV-E. Protocolo de validación empírica

Una validación posterior debe seleccionar un corpus de defectos reproducibles, estratificarlo por tipo y longitud, y ejecutar cada reductor varias veces cuando exista aleatoriedad. El hardware, tiempo máximo y política de caché deben mantenerse constantes. Para cada caso se recomienda verificar manualmente o mediante pruebas de regresión que la propiedad se conserva en el resultado final. Las comparaciones deben ser pareadas porque los mismos casos se procesan con todas las estrategias.

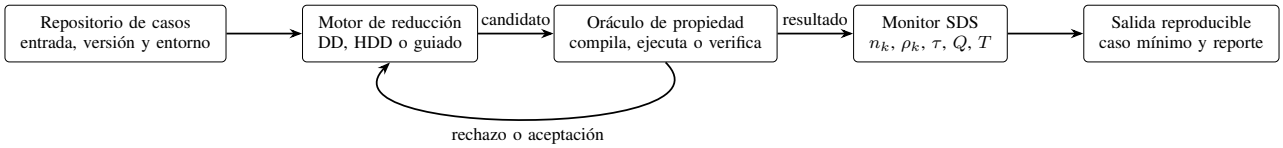


Figura 4. Arquitectura funcional de una herramienta de reducción con monitor de convergencia SDS.

Cuadro V
REQUISITOS PARA UNA HERRAMIENTA SDS DE REDUCCIÓN.

Requisito	Descripción
Preservación	Aceptar x' solo cuando el oráculo confirma la propiedad.
Trazabilidad	Registrar estados aceptados, candidatos rechazados y métricas.
Reproducibilidad	Fijar versiones, entorno, semilla y límites de ejecución.
Seguridad	Ejecutar candidatos en aislamiento con cuotas de recursos.
Extensibilidad	Incorporar transformaciones textuales, arbóreas y gramaticales.
Caché	Evitar reevaluar entradas idénticas o canónicamente equivalentes.
Visualización	Mostrar órbitas, ganancias marginales y punto de estabilización.
Exportación	Producir caso mínimo, script del oráculo y reporte técnico.

Además de las métricas usadas en la simulación, conviene medir la proporción de candidatos inválidos, el número de reinicios, el consumo máximo de memoria y la complejidad estructural final. Los resultados deben publicarse con datos por caso, no únicamente con promedios, a fin de permitir análisis alternativos y replicación. Un conjunto de casos donde un algoritmo no termina debe tratarse mediante análisis censurado o límites de tiempo explícitos, evitando sustituir arbitrariamente valores faltantes.

V. DISCUSIÓN DE RESULTADOS

Los resultados respaldan la utilidad del SDS como marco de análisis de reducción automática. La Proposición 1 explica por qué un operador que preserva la falla y reduce estrictamente una medida bien fundada debe terminar. Esta propiedad no garantiza optimalidad global: un punto fijo solo es irreducible respecto de las transformaciones y el orden de búsqueda disponibles. La distinción coincide con el concepto de 1-minimalidad de Delta Debugging y con la motivación de reductores especializados que amplían el espacio de transformaciones [1], [3].

La superioridad simulada de la estrategia guiada por sintaxis es coherente con el principio de evitar candidatos estructuralmente inválidos. HDD utiliza árboles para reducir por niveles y puede disminuir consultas frente a la reducción textual [2], [5]. Perses restringe las transformaciones a estructuras permitidas por una gramática y reporta mejoras sobre HDD en calidad y tiempo en diversos lenguajes [4]. La simulación no reproduce implementaciones concretas, pero sus parámetros representan esas diferencias teóricas: menor coeficiente de contracción y menor piso para el modelo guiado. Dado que estos perfiles fueron parametrizados para representar comportamientos diferenciados, las diferencias estadísticas observadas deben interpretarse como validación del marco SDS bajo condiciones simuladas y no como evidencia empírica de superioridad entre reductores reales.

El gran efecto observado en iteraciones ($W = 0{,}970$) indica que la forma de la trayectoria puede ser más discriminante que el tamaño final aislado. Dos reductores podrían obtener resultados de tamaño semejante, pero uno puede permanecer muchas rondas en una cola de mejoras marginales. El perfil (n_k) permite detectar estancamiento, estimar el costo restante y definir políticas de detención. En una herramienta real, podría interrumpirse la reducción cuando la ganancia relativa $1 - n_{k+1}/n_k$ permanezca bajo un umbral durante varias rondas.

Las consultas al oráculo son particularmente importantes porque cada candidato requiere compilar, ejecutar, simular o verificar una propiedad. En escenarios con pruebas costosas, el ahorro de consultas puede superar el beneficio de unos pocos tokens adicionales. C-Reduce combina transformaciones especializadas y ejecución paralela precisamente porque la reducción de programas reales exige equilibrar calidad y costo [3]. Extensiones modernas de HDD y métodos con reconstrucción de dependencias buscan aceptar transformaciones más informativas y reducir retrocesos [6], [8].

La interpretación mediante cuencas de atracción abre una línea adicional. Distintos casos que representan el mismo defecto podrían converger a puntos fijos iguales o equivalentes. Una cuenca grande sugeriría que el defecto posee un núcleo causal estable, mientras que múltiples atractores podrían indicar diversos patrones mínimos de reproducción. Esta hipótesis requiere definir una relación de equivalencia semántica entre casos reducidos, pues la igualdad de longitud no implica igualdad de causa.

Desde el punto de vista estadístico, la asimetría intensa del tiempo y de las consultas justifica pruebas no paramétricas. En evaluaciones de algoritmos, la distribución de tiempos suele presentar colas y valores extremos; por ello, la mediana, el RIQ, los diseños pareados y los tamaños de efecto son preferibles a conclusiones basadas únicamente en promedios [12]. La corrección de Holm controla el error familiar sin la rigidez de Bonferroni [18].

La principal limitación es el carácter sintético de los datos. Los resultados inferenciales demuestran que el protocolo estadístico detecta diferencias implantadas por el modelo, no que una herramienta específica supere empíricamente a otra. La validación externa debe usar casos reales de compiladores, navegadores, analizadores, APIs y sistemas ciberfísicos, con oráculos reproducibles. También debe considerarse la no determinación: fallas intermitentes pueden violar el supuesto $P(x) \in \{0, 1\}$ estable y requerir oráculos probabilísticos con ejecuciones repetidas [19].

VI. Validez, Reproducibilidad y Transferencia

VI-A. Amenazas a la validez interna y de conclusión

La simulación conserva el emparejamiento por caso y emplea un factor de dificultad compartido, lo que permite atribuir las diferencias implantadas principalmente a los parámetros de estrategia. Sin embargo, las ecuaciones generadoras fueron diseñadas para producir perfiles ordenados y, por tanto, los valores p no deben interpretarse como evidencia independiente sobre reductores reales. El objetivo inferencial consiste en verificar que el procedimiento propuesto detecta diferencias dinámicas y calcula tamaños de efecto coherentes. Una replicación empírica deberá preregistrar hipótesis, métricas y reglas de exclusión antes de observar resultados.

Otra amenaza es la dependencia entre métricas. El número de consultas influye directamente en el tiempo y se relaciona con las iteraciones; por ello, seis contrastes no equivalen a seis evidencias independientes. La corrección de Holm reduce el riesgo de falsos positivos dentro de cada familia de comparaciones, pero no elimina la redundancia conceptual. Se recomienda complementar los análisis univariados con modelos multivariantes o con un frente de Pareto, especialmente cuando una estrategia reduce más pero consume mucho más tiempo.

La discretización de tamaños e iteraciones genera empates, lo cual afecta la distribución exacta de Wilcoxon. En una evaluación real conviene conservar datos sin redondeos cuando corresponda, informar la cantidad de ceros y empates, y utilizar métodos exactos o permutacionales si el tamaño muestral lo permite. También se recomienda presentar intervalos de confianza mediante remuestreo, además de valores p, para cuantificar la incertidumbre de medianas y diferencias.

VI-B. Validez de constructo y estado dinámico

La longitud es una observable útil, pero incompleta. Un programa de 30 tokens puede ser más complejo que otro de 50 si contiene dependencias, macros o construcciones semánticas densas. Del mismo modo, dos entradas de igual longitud pueden activar rutas distintas. Por esta razón, el SDS completo debe definirse en $\mathcal{T}_P$ y no únicamente en $\mathbb{N}$. La proyección escalar es apropiada para visualizar contracción, pero no reemplaza la identidad estructural del estado.

Una extensión natural usa un vector

$$z_k = (n_k, d_k, v_k, q_k), \tag{9}$$

con longitud n_k, profundidad sintáctica d_k, número de dependencias v_k y costo acumulado q_k. El operador actúa entonces en un espacio multidimensional y puede reducir una componente mientras aumenta otra. El análisis requiere órdenes parciales, dominancia de Pareto o funciones de energía ponderadas. Esta ampliación permitiría distinguir reductores que producen casos cortos pero estructuralmente complejos de aquellos que preservan claridad y legibilidad.

El predicado de fallo también puede ser imperfecto. Un oráculo basado en coincidencia textual de mensajes podría aceptar casos que provocan una causa distinta. Se recomienda definir una firma de fallo que combine código de salida, ubicación, traza, aserción y condiciones del entorno. Para fallas intermitentes, el estado debería incorporar una probabilidad estimada de reproducción y el oráculo debería repetir ejecuciones hasta alcanzar una precisión definida [19].

Cuadro VI
Lista de comprobación para una replicación empírica.

Elemento	Información requerida
Corpus	Origen, licencias, tamaño y tipos de fallos.
Oráculo	Script, repetibilidad, tiempo límite y firma de fallo.
Reductores	Versión, configuración, gramática y transformaciones.
Entorno	Hardware, sistema operativo, compiladores y dependencias.
Ejecución	Semillas, repeticiones, paralelismo y política de caché.
Métricas	Unidad de longitud, tiempo, consultas, memoria y validez.
Estadística	Hipótesis, normalidad, pruebas, correcciones y efectos.
Artefactos	Datos por caso, código, resultados y documentación.

VI-C. Validez externa y transferencia tecnológica

Los coeficientes de contracción y pisos usados en la simulación no son universales. Cambian con el lenguaje, la gramática, el defecto, el reductor y el costo del oráculo. Una evaluación externa debe incluir entradas de diferentes tamaños y dominios, y no limitarse a casos donde todas las estrategias concluyen. Los tiempos máximos, errores de herramienta y resultados no reducibles son parte del desempeño y deben conservarse en el análisis.

La transferencia a una organización requiere integrar la herramienta con su sistema de incidencias y repositorio de pruebas, pero el valor principal no depende de una tecnología específica. El reporte SDS puede adjuntarse a una incidencia e incluir: entrada original; entrada reducida; script del oráculo; gráfica de trayectoria; consultas; tiempo; versión del producto; y advertencias sobre minimalidad. De este modo, el equipo receptor puede reproducir el fallo sin reconstruir el entorno de reducción.

En proyectos de código abierto, los casos reducidos disminuyen el esfuerzo de revisión y evitan divulgar archivos completos que puedan contener información irrelevante. No obstante, reducir no garantiza anonimización: un fragmento mínimo todavía puede contener secretos o datos personales. Antes de compartirlo deben aplicarse controles de seguridad, clasificación de información y revisión humana.

VI-D. Lista de comprobación para replicación

El Cuadro VI propone una lista mínima para estudios posteriores. Su finalidad es prevenir comparaciones no reproducibles y separar claramente los costos del oráculo, del reductor y del entorno.

El cumplimiento de esta lista permite transformar la prueba de concepto en un estudio de Ingeniería de Sistemas verificable. Además, facilita comparar resultados entre instituciones y construir un repositorio de trayectorias que permita estudiar regularidades, predecir tiempos de convergencia y entrenar selectores automáticos de estrategia.

VII. Conclusiones y Recomendaciones

VII-A. Conclusiones

El SDS propuesto representa la reducción automática como una dinámica sobre el espacio de casos que preservan una falla y utiliza la longitud como observable. Esta formulación permite interpretar la sucesión de candidatos como una órbita, el resultado irreducible como punto fijo, el número de rondas como tiempo de llegada y los grupos de casos equivalentes como posibles cuencas de atracción.

Se demostró que todo operador estrictamente reductor respecto de una longitud natural termina en tiempo finito. Asimismo, una desigualdad afín de contracción proporciona una cota explícita para la trayectoria y permite estimar la región de estabilización.

En el experimento computacional simulado, las pruebas de Shapiro–Wilk justificaron el uso de métodos no paramétricos. Friedman encontró diferencias significativas en tamaño final, iteraciones, consultas y tiempo, con efectos grandes. La estrategia guiada por sintaxis presentó las menores medianas, seguida de HDD y DD. Se aceptaron las cuatro hipótesis específicas y, en conjunto, la hipótesis general.

Los resultados no constituyen evidencia comparativa definitiva sobre herramientas reales. Su contribución es metodológica: muestran cómo diseñar indicadores, pruebas y visualizaciones para evaluar reductores desde una perspectiva dinámica y reproducible.

VII-B. Recomendaciones

Se recomienda replicar el estudio con corpus públicos de fallos y ejecutar implementaciones reales de DD, HDD, C-Reduce y Perses bajo el mismo hardware y oráculo. Deben reportarse tamaño inicial y final, consultas, tiempo, tasa de candidatos inválidos, memoria y estabilidad entre repeticiones.

Conviene extender el estado del SDS más allá de la longitud, incorporando número de nodos, profundidad sintáctica, dependencias, cobertura y costo del oráculo. Un estado vectorial permitiría estudiar dominancia de Pareto entre calidad y costo, evitando reducir el fenómeno a una sola observable.

Finalmente, se recomienda analizar oráculos probabilísticos para fallas intermitentes, estudiar equivalencia semántica entre atractores y desarrollar reglas de detención adaptativas basadas en la ganancia marginal de reducción.

Referencias

[1] A. Zeller and R. Hildebrandt, "Simplifying and isolating failure-inducing input," *IEEE Trans. Software Eng.*, vol. 28, no. 2, pp. 183–200, 2002, doi: 10.1109/32.988498.

[2] G. Misherghi and Z. Su, "HDD: Hierarchical delta debugging," in *Proc. 28th Int. Conf. Software Engineering*, 2006, pp. 142–151, doi: 10.1145/1134285.1134307.

[3] J. Regehr, Y. Chen, P. Cuoq, E. Eide, C. Ellison, and X. Yang, "Test-case reduction for C compiler bugs," in *Proc. 33rd ACM SIGPLAN Conf. Programming Language Design and Implementation*, 2012, pp. 335–346, doi: 10.1145/2254064.2254104.

[4] C. Sun, Y. Li, Q. Zhang, T. Gu, and Z. Su, "Perses: Syntax-guided program reduction," in *Proc. 40th Int. Conf. Software Engineering*, 2018, pp. 361–371, doi: 10.1145/3180155.3180236.

[5] R. Hodován and Á. Kiss, "Modernizing hierarchical delta debugging," in *Proc. 7th Int. Workshop Automating Test Case Design, Selection, and Evaluation*, 2016, pp. 31–37, doi: 10.1145/2994291.2994296.

[6] D. Vince, R. Hodován, D. Bársony, and Á. Kiss, "Extending hierarchical delta debugging with hoisting," in *Proc. 15th IEEE Conf. Software Testing, Verification and Validation*, 2022, pp. 134–145.

[7] D. Vince, R. Hodován, and Á. Kiss, "Iterating the minimizing delta debugging algorithm," in *Proc. 22nd IEEE Int. Conf. Software Quality, Reliability and Security*, 2022, pp. 666–677.

[8] Q. Feng, X. Ma, Y. Tian, W. Song, and P. Liang, "DRReduce: Enhancing syntax-guided program reduction with dependency reconstruction," arXiv:2605.19412, 2026.

[9] K. T. Alligood, T. D. Sauer, and J. A. Yorke, *Chaos: An Introduction to Dynamical Systems*. New York, NY, USA: Springer, 1996.

[10] R. L. Devaney, *An Introduction to Chaotic Dynamical Systems*, 2nd ed. Boulder, CO, USA: Westview Press, 2003.

[11] D. Lind and B. Marcus, *An Introduction to Symbolic Dynamics and Coding*. Cambridge, U.K.: Cambridge Univ. Press, 1995.

[12] A. Arcuri and L. Briand, "A hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering," *Software Testing, Verification and Reliability*, vol. 24, no. 3, pp. 219–250, 2014, doi: 10.1002/stvr.1486.

[13] C. R. Harris *et al.*, "Array programming with NumPy," *Nature*, vol. 585, pp. 357–362, 2020, doi: 10.1038/s41586-020-2649-2.

[14] P. Virtanen *et al.*, "SciPy 1.0: Fundamental algorithms for scientific computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020, doi: 10.1038/s41592-019-0686-2.

[15] S. S. Shapiro and M. B. Wilk, "An analysis of variance test for normality (complete samples)," *Biometrika*, vol. 52, nos. 3–4, pp. 591–611, 1965.

[16] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," *J. Amer. Statistical Assoc.*, vol. 32, no. 200, pp. 675–701, 1937.

[17] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

[18] S. Holm, "A simple sequentially rejective multiple test procedure," *Scandinavian J. Statistics*, vol. 6, no. 2, pp. 65–70, 1979.

[19] P. Valle, S. Ali, and A. Arrieta, "Delta debugging for cyber-physical systems with flaky test executions," arXiv:2607.25695, 2026.

[20] R. Hodován, Á. Kiss, and T. Gyimóthy, "Coarse hierarchical delta debugging," in *Proc. IEEE Int. Conf. Software Maintenance and Evolution*, 2017, pp. 194–203.

[21] M. R. I. Rabin, A. Hussain, and M. A. Alipour, "Syntax-guided program reduction for understanding neural code intelligence models," arXiv:2205.14374, 2022.

[22] G. Gharachorlu and W. N. Sumner, "One-pass delta debugging," in *Proc. IEEE Int. Conf. Software Testing, Verification and Validation*, 2021, pp. 136–147.

[23] Á. Kiss, R. Hodován, and T. Gyimóthy, "Tree-based test case reduction," in *Advances in Computers*, vol. 111. Amsterdam, The Netherlands: Elsevier, 2018, pp. 133–165.

[24] A. Zeller, "Simplifying and isolating failure-inducing input: A retrospective on Delta Debugging," *IEEE Trans. Software Eng.*, vol. 51, no. 3, pp. 1–16, 2025.