A Multimodal Intelligent Change Assessment Framework for Microservice Systems Based on Large Language Models

Yongqian Sun
Nankai University
Tianjin, China

Tinghua Zheng
Nankai University
Tianjin, China

Xidao Wen
Tsinghua University
Beijing, China

Weihua Kuang
Nankai University
Tianjin, China

Heng Liu
CHINA TIANCHEN ENGINEERING CORPORATION LTD.
Tianjin, China

Shenglin Zhang
Nankai University
Tianjin, China

Chao Shen
Nankai University
Tianjin, China

Bo Wu
Tencent Technologies
Beijing, China

Dan Pei
Tsinghua University
Beijing, China

Abstract

Frequent changes in large-scale online service systems often lead to failures, threatening system reliability. To overcome the limitations of existing techniques in erroneous change detection, failure triage, and root cause change analysis, this paper presents a multimodal intelligent change assessment framework based on large language models. Our framework integrates retrieval-augmented generation techniques and leverages unified representation of multimodal data, enhanced knowledge access, and domain-specific LLMs to automate the entire change management lifecycle. Experiments on two microservice system datasets show that our method outperforms state-of-the-arts in accuracy, efficiency, and minimizing manual intervention. This work provides a robust solution for change management and valuable insights into improving system stability and optimizing operational workflows.

Keywords

Software Change, Anomaly Detection, Failure Triage, Root Cause Analysis, LLMs, RAG.

ACM Reference Format:
Yongqian Sun, Tinghua Zheng, Xidao Wen, Weihua Kuang, Heng Liu, Shenglin Zhang, Chao Shen, Bo Wu, and Dan Pei. 2025. A Multimodal Intelligent Change Assessment Framework for Microservice Systems Based on Large Language Models. In *Companion Proceedings of the 33rd ACM Symposium on the Foundations of Software Engineering (FSE '25), June 23–27, 2025, Trondheim, Norway*. ACM, New York, NY, USA, 11 pages. https://doi.org/XXXXXXX.XXXXXXX

ESEC/FSE'25, June 23–27, 2025, Trondheim, Norway

ACM ISBN 978-1-4503-XXXX-X/25/06
https://doi.org/XXXXXXX.XXXXXXX

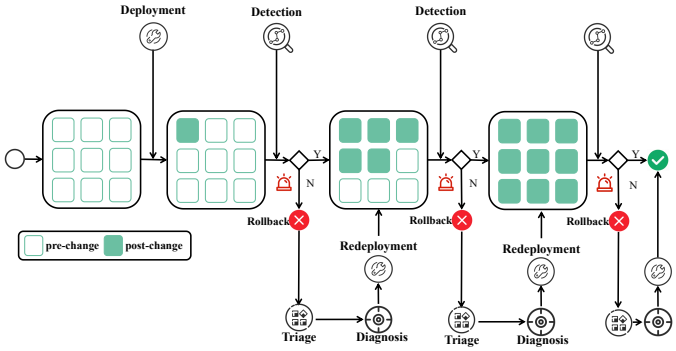

Figure 1: A Typical Software Change Life Cycle. When an erroneous change occurs, multiple rollback deployments are required to complete the change.

1 Introduction

In large-scale online systems, a large number of software changes occur daily, leading to significant risks of service failures. For instance, studies reveal that major online service providers like Google implement over 10,000 software changes each day [27]. Another study reports that approximately 70% of service incidents stem from erroneous software changes [1]. At Baidu, 54% of service failures are attributed to changes [40]. Such failures can result in severe economic losses; for example, Facebook outage caused by an erroneous software change led to an estimated $60 million loss in 2021 [32]. These examples underscore the critical need for timely detection of erroneous software changes, accurate diagnosis of their root causes, and rapid resolution to mitigate business impact.

The software change life cycle consists of several stages: (1) Deployment: Developers deploy a new software version to a limited set of servers, virtual machines, or containers, monitoring its status; (2) Detection: If anomalies occur, the deployment is halted and rolled back; (3) Triage: The incident is assigned to the appropriate engineering team after an initial assessment; (4) Diagnosis: Engineers investigate the root cause, which often requires multiple rounds of communication; (5) Redeployment: Once resolved, the change

is redeployed. A typical software change life cycle is illustrated in Figure 1.

Handling software change errors effectively involves three primary steps: erroneous change detection (ECD), failure triage (FT), and root cause change analysis (RCCA). These tasks are essential for operation and maintenance management. ECD has received considerable attention, with work like SCWarn [43], Kontrast [31], Lumos [24], and Funnel [40] automating this step. However, FT and RCCA remain largely manual, with limited automation. While ChangeRCA [38] addresses RCCA, no mature solutions currently exist for automating FT. When an erroneous change is detected by ECD, the change is often rolled back immediately, leaving FT and RCCA to be performed manually by engineers, who must analyze multimodal data such as metrics and logs. This manual intervention increases both labor costs and operational delays.

Despite the advancements in ECD automation, no existing method fully integrates all three tasks—ECD, FT, and RCCA—into a unified framework. In practice, engineers rely on separate single-task techniques for each stage, resulting in redundant efforts (e.g., feature extraction, model training, data organization), which increases the time needed to resolve erroneous changes [34].

To this end, we propose SCELM (**S**oftware **C**hange **E**valuation and **L**ifecycle **M**anagement), to automate and streamline the critical stages of change management. SCELM integrates large language models (LLMs) and multi-task learning to combine these tasks into a single, automated pipeline. Unlike existing methods that require separate, manual processes, SCELM streamlines the entire workflow—from detecting erroneous changes and triaging incidents to identifying root causes. Using a Retrieval-Augmented Generation (RAG) model, SCELM improves the ability of LLMs to access and update operational knowledge, ensuring that it adapts to evolving scenarios. This unified approach enhances efficiency, reduces labor costs, and accelerates issue resolution, ultimately leading to more reliable and cost-effective service management.

The contributions of our work are summarized as follows:

- SCELM is the first framework to integrate ECD, FT, and RCCA into a unified pipeline. By leveraging LLMs with multi-task learning capabilities, SCELM automates the entire software change assessment process, offering a seamless solution for change management.
- We designed an RAG model to automate the three key steps in change assessment. This model enhances LLMs' ability to access and process operational knowledge, enabling more accurate change-related semantic information management. SCELM dynamically updates and manages knowledge bases, significantly increasing its practical applicability.
- To substantiate the effectiveness of SCELM, we have implemented SCELM on a large Software as a Service (SaaS) microservice system for more than 11 months, monitoring thousands of changes per week.
- Extensive experiments were conducted on two microservice system datasets, comprising 54 and 364 instances, respectively. The results demonstrate the effectiveness and efficiency of SCELM . Our source code is publicly available[1].

[1]https://anonymous.4open.science/r/AIOps-SCELM

2 Motivation

2.1 Inefficiencies in Change Management

Change management is critical in digital operations, ensuring rapid failure detection and resolution to minimize disruptions and enhance reliability. However, current practices face challenges like the lack of standardized frameworks and the high cost of manual involvement. While techniques such as ECD, FT, and RCCA are widely applied, they remain isolated and are not yet part of a unified, automated framework.

Currently, On-Call Engineers (OCEs) oversee the entire change lifecycle, from planning and rollout to issue resolution. When erroneous changes occur, OCEs face a heavy workload, leading to inefficiency and potential errors. According to our experience with OCEs, diagnosing a failure typically takes 2–3 hours, involving detailed analysis, team discussions, and formal reviews. This process is time-consuming and costly.

For example, during a service rollout, each system node is updated sequentially. If an erroneous change occurs, engineers must correlate change history, metrics, logs, and environmental conditions. Ideally, this would take 30 minutes to 1 hour to meet Service Level Agreement (SLA) targets. However, manual workflows often delay this, complicating efforts to maintain service continuity.

To address these challenges, an automated, end-to-end change management framework is necessary. Integrating ECD, FT, and RCCA into a cohesive system can streamline the process, reduce reliance on manual intervention, and improve efficiency.

2.2 Improving Change Management with LLMs

Managing frequent software changes is a significant challenge for OCEs. To handle these changes efficiently, an automated solution is needed to reduce manual work and speed up decision-making. LLMs are well-suited for this, excelling at tasks like anomaly detection, failure triage, and root cause analysis, which require processing large amounts of unstructured data [26].

Fine-tuning models for change assessment requires large, labeled datasets and substantial computational resources, making it unsuitable for real-time tasks, especially in data-scarce and time-sensitive environments [7]. RAG, however, dynamically retrieves external knowledge, making it more efficient and adaptable for real-time change assessment, even with smaller datasets.

LLMs can generate incorrect or nonsensical responses, known as hallucinations[4]. RAG mitigates this by enhancing the LLM's access to relevant external knowledge, improving the accuracy and reliability of real-time responses. RAG enables LLMs to "search" external knowledge during inference, much like an "open-book exam" [41]. This is valuable in change management, where past experiences are crucial for accurate decision-making.

3 Problem Formulation

Directly inputting multimodal data into LLMs for change assessment management encounters the following challenges:

3.1 Complexities in Handling Multimodal Data

In change assessment tasks, the data involved is not only large in volume but also comes from multiple sources and in varying formats.

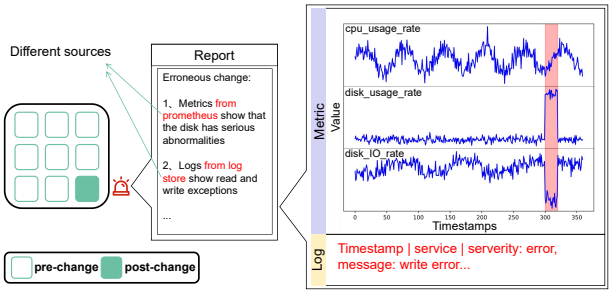

Figure 2: An erroneous change Case Presented from Different Sources (metrics, logs). Metrics are time series data and logs are semi-structured text data.

These include structured data (e.g., metrics like CPU usage, response time), semi-structured data (e.g., service logs), and unstructured data (e.g., fault reports and change notes). Each of these data types has different characteristics, structures, and relationships, which complicates their integration into a single representation. Different sources data are illustrated in Figure 2.

Current methods for handling multimodal data, such as SCWarn [43] and ANOFusion [39], typically convert these diverse data types into a unified format, such as time series or metrics. However, such conversion often leads to the loss of crucial semantic information embedded within the data, especially in logs where operational context and historical events are significant. This oversimplification limits the model's ability to perform accurate analysis. Therefore, the core challenge is how to represent this multimodal data in a way that retains as much semantic detail as possible, while still making it suitable for machine learning models, particularly LLMs.

Moreover, LLMs, particularly smaller models, are not able to process multimodal data directly[2, 25]. These models are typically limited to processing text-based input, while more complex, multimodal data (including both logs and metrics) cannot be effectively processed without advanced pre-processing. This mismatch between data types and model capabilities creates a significant barrier for real-time change assessment tasks.

3.2 Hallucinations in LLMs' Generated Content

While LLMs have advanced significantly in natural language processing (NLP), both academic and industrial communities recognize their limitations. A major issue is their tendency to generate factually incorrect or nonsensical outputs, known as "hallucination." This poses risks to LLM deployment, as their knowledge base is static and based on training data, limiting their ability to provide accurate answers for recent developments or complex, domain-specific problems. This is referred to as "intrinsic hallucination."

To mitigate these issues, recent studies have introduced RAG to enhance LLM capabilities[13, 15, 28]. However, RAG is not immune to hallucinations. Low-quality or irrelevant documents retrieved during the retrieval phase can introduce errors, leading to inaccurate responses, a phenomenon known as "extrinsic hallucination" [4].

In real-world environments, engineers often deal with complex, recurring anomalies. Effective resolution depends on their expertise and understanding of the system. However, LLMs struggle to handle novel anomalies or new scenarios, limiting their usefulness in dynamic, high-stakes environments.

3.3 Limitations of LLMs in Change Management

Change management is a complex task involving system dependency analysis, real-time monitoring, change impact assessment, and contingency planning. This requires a deep understanding of system architecture, the operating environment, historical data, and interaction effects—areas where LLMs often struggle.

Traditional tools in operational analysis rely on predefined rules, algorithms, or expert knowledge to quickly identify bottlenecks and risks. For example, before implementing a change, automated tools can assess its impact and generate detailed risk reports. While general LLMs excel in language understanding, they often rely on patterns and correlations in training data. If the data lacks domain-specific knowledge or real-time updates, the generated content may be inaccurate or incomplete.

For instance, in change impact assessments, LLMs may generate reasonable-sounding suggestions that don't align with the system's operational logic. Complex system dependencies and interaction effects are often overlooked, leading to errors. A specific example is database architecture adjustments, where LLMs may fail to identify critical cross-database dependencies, leading to downstream impacts like data inconsistencies, service interruptions, or cascading failures. Thus, while LLMs are strong in language processing, their effectiveness is limited in complex operational tasks [14].

3.4 Problem Description

To address data confidentiality concerns and the high costs of commercial LLMs, we employ open-source models. Given the real-time and responsive requirements of change management, we adopt small-parameter LLMs (e.g., 7B) to balance processing speed and computational efficiency. Combined with RAG, these models enable dynamic knowledge retrieval from external sources, facilitating efficient handling of tasks like ECD, FT, and RCCA.

This approach leverages RAG-enhanced small-parameter LLMs to deliver real-time responses and accurate operational issue analysis. By simulating OCEs' cognitive processes, we aim to establish a unified LLM-based framework for processing multimodal data, extracting actionable insights, and delivering timely assessments. This reduces manual intervention while improving the accuracy and speed of failure management, addressing the critical demands of live system change management.

4 Approach

4.1 Overview

We propose SCELM, a change assessment framework for ECD, FT, and RCCA tasks. As shown in Figure 3, SCELM operates in two stages: offline preparing and online stage. The offline preparing stage constructs a historical change knowledge base by modeling past change orders, logs, and metrics. The online stage processes real-time data, using RAG to support efficient ECD, FT, and RCCA.

Specifically, SCELM consists of three modules. Module 1 synthesizes data from metrics and logs into domain-specific text using preprocessing and hybrid algorithms, enabling consistent analysis. Module 2 builds and vectorizes a historical change knowledge

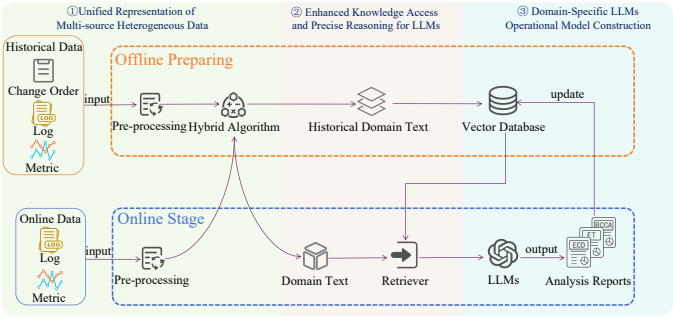

Figure 3: The Overall Framework of SCELM. Multimodal data are converted into domain text and then fed to LLMs to generate analysis reports.

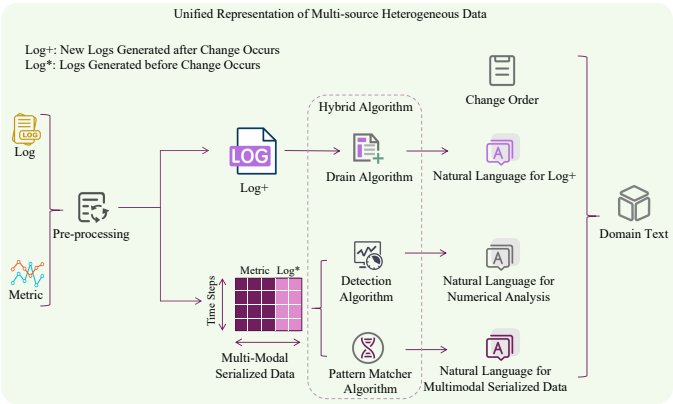

Figure 4: The Framework Diagram of Module 1. Metrics and logs are converted into natural language after pre-processing and hybrid algorithm, and then converted into domain text in combination with change order information.

base to mitigate hallucination issues, enhancing LLMs reasoning and change assessment accuracy. Module 3 employs RAG to simulate OCEs cognitive processes, producing actionable reports that analyze failure types, identify root causes, and provide targeted recommendations.

4.2 Unified Representation of Multimodal Data

This module enhances the identification and classification of failure points by analyzing data derived from ECD. It extracts information about failure-related anomalies and change order data, providing a targeted textual summary for individual cases. The framework diagram for this module is shown in Figure 4.

4.2.1 Multimodal Data Serialization and Log Processing. To process the various types of data involved in change assessments, we serialize multimodal data (metrics and logs) into time series. Metrics are processed by standard normalization techniques, while logs are handled by the approach in SCWarn [43]. Specifically, we calculate the frequency of parsed log templates using Drain [8] to convert logs into time series.

In typical conditions, logs usually match existing templates. However, in erroneous change situations, the volume of new logs increases, resulting in new log templates that are not matched by existing ones. The SCWarn approach handles this by assessing the number of new logs that do not match any template. However, our

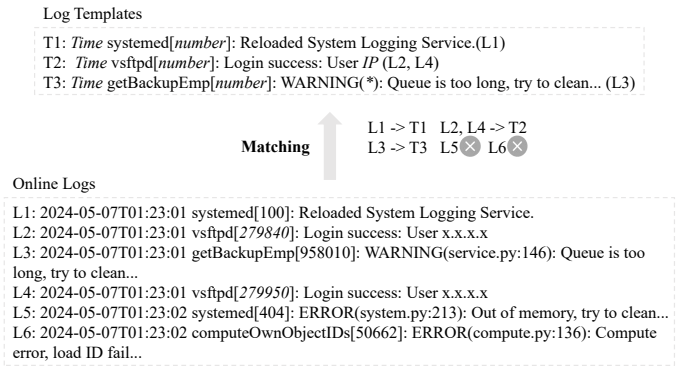

Figure 5: Log Matching Example. L5 and L6 are new logs generated after the change occurs and cannot match the log template before the change.

observations in real-world scenarios indicate that new log templates arising from changes often contain significant semantic information, which is crucial for diagnosing failures and performing root cause analysis.

As shown in Figure 5, L5 and L6 represent new log templates generated after a change. These logs are unmatchable with existing templates but provide critical failure insights—L5 signals system failure, and L6 indicates business failure. From a semantic perspective, these logs directly point to the failure cause. If we only focus on the number of new logs (as SCWarn does), we would miss this important semantic content. After pre-processing, these logs are transformed into time series and integrated with the metrics data. New log templates, due to their semantic richness, are represented in natural language for cohesive processing by LLMs.

4.2.2 Enhanced Identification and Classification of Failure Points. Anomalies in time series data can be classified based on their duration and shape. When anomalies are detected, OCEs typically compare them with historical incidents to understand the nature of the anomaly and derive troubleshooting recommendations. We enhance this process by using a pattern matcher [33] to define graphical rules and patterns that identify the shapes of anomalies. This approach helps classify anomalies as either transient fluctuations or persistent anomalies.

For example, a "single spike" in response time may indicate a temporary issue that quickly resolves, while a "steady increase" in memory usage could signal a more persistent problem. By classifying anomalies in this way, we can filter out irrelevant noise and focus on the issues that matter most. As shown in Figure 6, these anomaly shapes are converted into natural language descriptions through the pattern matcher, making it easier for LLMs to process and interpret the results.

4.2.3 Integration of Anomaly Metric Data and Change Order Information. We integrate the multimodal serialized data with the change order information to produce a domain text that consolidates data from the previous steps and encapsulates the key insights needed for LLMs to generate accurate change assessments.

The purpose of the domain text is to structure diverse data types (e.g., time-series metrics, log data, change order information) in a way that is understandable by LLMs, facilitating more accurate change analysis. The domain text includes the following elements:

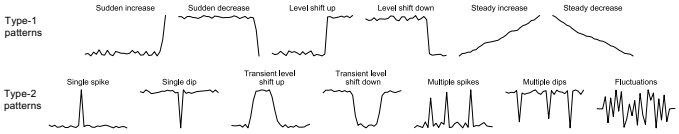

Figure 6: Typical Anomaly Patterns Summarized From Large-scale Real-world Data. Type-1: still in abnormal state. Type-2: recover to normal state.

(1) **Change order records** that include the change order ID, the affected service, type of the change (e.g., change configurations), and the start and end times for submission and analysis.
(2) **Timestamps of anomalies identified** that show when anomalies were detected by the system, linking changes to observed issues.
(3) **Anomaly classifications and metric descriptions** that define the types of anomalies and describes the related metrics
(4) **Metric comparisons before and after the change** that compare key metrics before and after the change, summarizing their impact.
(5) **Detailed comparison of metrics and findings** that provide a comparison of individual metrics (e.g., max, min, average) before and after the change, along with a summary of the results.
(6) **New log template descriptions** that describe new log templates created after the change, which may highlight new failures or changes.

Historical change orders typically contain complete information, including the status of previous changes, whether those changes were erroneous, the types of failures that occurred, the root causes, and the corresponding solutions. This historical data is invaluable in constructing a historical experience knowledge base that informs the analysis of future changes. On the other hand, online change orders provide real-time change-related information and help streamline the acquisition of multimodal data, making it easier to process and generate real-time assessments.

By integrating these data points into a cohesive domain text, we ensure that LLMs have a structured, comprehensive input for generating accurate results in tasks like ECD, FT, and RCCA.

4.3 Enhanced Knowledge Access and Precise Reasoning for LLMs

As shown in Figure 7, Module 2 is responsible for building and leveraging a knowledge base of historical experience to improve the understanding and response capabilities of LLM in change scenarios. This module enhances the model's ability to analyze changes by providing it with extensive contextual information and historical instances. By incorporating past change records, analysis results, and exception-handling experiences, LLMs can draw from past knowledge during new change analysis, improving the accuracy of the generated text. This process helps LLMs identify change types, recognize abnormal characteristics, and propose possible response strategies, ultimately offering OCEs more instructive analysis recommendations. The module consists of three steps:

4.3.1 Vectorization of Domain-Specific Text. We vectorize the domain-specific text of each change case to help LLMs quickly retrieve and reference key information when for analysis. This process transforms text data into points in a high-dimensional vector space, preserving the semantic content. By doing so, LLMs can match context and integrate historical experience into new analyses. The vectorized texts can either serve as direct input for subsequent tasks or as an index for retrieving and matching similar historical cases, improving the overall efficiency of change evaluations.

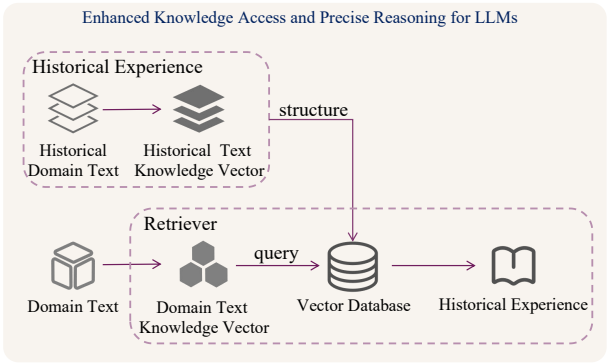

Figure 7: The Framework Diagram of Module 2. Retriever finds the most similar historical experience in the vector database based on the input domain text.

4.3.2 Domain Text Knowledge Point Extraction. To better organize and utilize historical experience, we implement a metric clustering method. This method classifies and summarizes various metrics based on their physical meaning as knowledge points in domain texts. Metrics showing significant abnormal fluctuations or trends are prioritized and labeled as Top Abnormal Key Performance Indicators (KPIs). In practice, these KPIs often represent the status of critical system components or nodes, and they carry significant physical implications. For instance, multiple related metrics concerning server anomalies can help identify the root cause of failures. Metric clustering standardizes the classification of these metrics, making anomaly detection more systematic and robust.

4.3.3 Building a Knowledge Base and Loading Historical Experience. The historical experience base is constructed from historical change orders, which contain information on whether a change was successful, the root cause of any failures, and the solutions. We retrieve relevant metrics and logs from these historical records, process with Module 1, and integrate into historical domain texts.

Once the historical experience has been vectorized and integrated, the system can search for similar historical cases within the knowledge base. By matching knowledge elements from the current change with similar past cases, the system retrieves additional relevant texts for deeper analysis. These texts are incorporated into the Prompt Enhancement Module, providing additional context and supporting the current analysis. The wealth of historical knowledge enables LLMs to quickly reference past experiences when analyzing new changes, aligning their output more closely with real-world conditions and minimizing potential biases. Furthermore, these texts help LLMs automatically identify potential risk points, common failure characteristics, and appropriate response measures, making the change analysis process more comprehensive.

The main objective of loading historical change experience is to continuously strengthen the change analysis capabilities of LLMs.

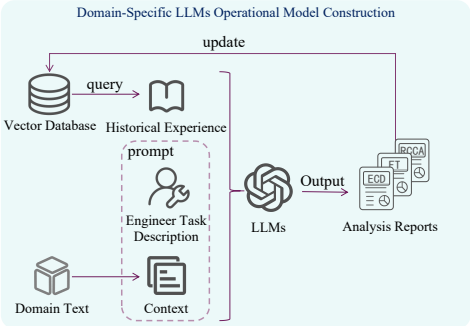

Figure 8: The Framework Diagram of Module 3. LLMs generates change analysis with prompts and historical experience.

By reusing and expanding past experiences, LLMs can form more consistent analysis logic for new changes, avoid repetitive errors, and improve the accuracy and consistency of generated results. Additionally, incorporating historical knowledge ensures that the generated analysis is aligned with the OCE's workflow, making the model's suggestions more practical and actionable, ultimately providing strong support for decision-making during system changes.

4.4 Domain-Specific LLMs Operational Model Construction

As shown in Figure 8, Module 3 focuses on generating detailed change assessment reports using LLMs. This module takes domain-specific text, system status, and historical experiences, then generates comprehensive and structured reports that provide clear guidance for change management. The goal is for LLMs to not only generate analysis content that fits the context but also offer professional responses based on realistic operational scenarios. The content generated includes judgments on whether a change is a failure, failure triage, root cause analysis, and proposed solutions. This module is mainly divided into two steps:

4.4.1 RAG Retriever Augment Generation. Retriever loads historical experience from the change management knowledge base. The process of loading historical records and experience data is detailed in Module 2. By utilizing past data trends, events, and results, this step enables LLMs to provide deeper, more professional responses beyond surface-level analysis. It allows LLMs to suggest more accurate change management strategies based on historical context. Augment task involves describing the task sequence and structuring the output.First, we model the logical sequence of tasks followed by OCEs during real-world scenarios. For example, for multi-step change tasks, the RAG framework outlines the process from initiation to final confirmation. This helps LLMs align their outputs with the logical workflows used by OCEs. Then, the generated analysis is organized into sections such as background introduction, change analysis, risk assessment, and result summary. This format enhances readability, helps users grasp essential content quickly, and supports subsequent system processing.

Generation task involves adaptive text generation, which tailors the output based on the characteristics of the specific change scenario. RAG adapts its generated content based on the context of the change[14]. With significant anomalies or fluctuations, it enhances the analysis with risk assessments and mitigation strategies. For less complex changes, the analysis is simplified. This adaptive generation ensures that the produced content is more context-appropriate, improving the accuracy and relevance of the analysis.

Change Ticket

NO.50004

Service	adservice	**Start time**	2024-05-07 19:30:00	**End time**	2024-05-08 19:30:00
Environment	BJ-k8s	**Container**	AD-env-cont	**Host**	x,y,z,1,2,3
Config	env=BJ-k8s, cont=AD-cont, host=6, ip=x,x,x,x				
Operation	Migrate adservice to new environment(BJ-k8s), ip=x,x,x,x ...				
Involved Data	Related 6 machine KPIs of host, 24 business KPIs of adservice, logs of adservice				

LLMs Analysis Report

Change Type: failure

Failure Triage: Wrong IP Address-Configuration Errors-Defective Software Changes

Top 5 abnormal kpi: cpu_usage, memory_usage, net_send_packet, net_recv_packet, istio_count

reason: The data shows significant changes in CPU usage, memory usage, network packet sending and receiving, and Istio count after the change. This indicates a potential issue with the system's performance or resource allocation.

solution: 1. Investigate Resource Utilization:Analyze system logs and resource monitoring tools to pinpoint the cause of decreased CPU and memory usage.

2. Network Performance Analysis:Conduct network diagnostics to identify bottlenecks or connectivity problems affecting 'net_recv_packet'.

3. Scale Resources: If necessary, increase server capacity or adjust load balancing configurations to accommodate the increased request volume reflected in 'istio_count'.

Figure 9: An Example Report Provided by SCELM.

4.4.2 Generating Analytical report. Once the domain texts and contextual information are provided, LLMs generate a detailed change management analysis report, covering all key tasks. For example, if anomalies and failures occur during the change process, the LLMs can categorize the failure type, identify the root cause, and provide targeted recommendations for resolution. The generated report includes the following sections:

(1) Change Type: Normal or failure.
(2) Failure Triage: If the change type is failure, provide the failure category of change.
(3) Top 5 Abnormal KPIs: If change type is failure, provide the 5 KPIs that have the largest impact on change.
(4) Reason: Provide the reason for this change of judgment.
(5) Solution: If change type is failure, provide specific solutions that may be useful.

For instance, as shown in Figure 9, an erroneous change during a service migration was caused by a configuration error—specifically an IP configuration issue. The LLMs' judgment is based on established reasoning, and the provided solutions can be quickly implemented by engineers to resolve the issue.

Through Module 3, LLMs generate structured, detailed change management analysis reports. The combination of RAG's retrieval and generation capabilities, along with adaptive generation, ensures that the analysis is contextually appropriate and actionable. This process provides a solid basis for decision support, improves transparency, and enhances the manageability of system changes. By offering well-structured analysis with clear steps, these reports facilitate effective decision-making and better risk management.

5 Evaluation

In this section, we address the following research questions:

- RQ1: How does SCELM perform in ECD, FT, and RCCA?
- RQ2: How does each component contribute to SCELM?
- RQ3: How do the major hyperparameters of SCELM impact its performance?
- RQ4: How does RAG reduce the impact of hallucinations and improve the expertise in the field of change management?
- RQ5: How does the size of LLM paramter influence performance?

Table 1: Datasets Descriptions.

Dataset	#Instances	#Failure	#Normal	#Failure Types	#Records
D1	54	40	14	5	metric 26,438,400 log 4,529,848,320
D2	364	183	181	6	metric 4,193,280

5.1 Experimental Setup

5.1.1 Datasets. To evaluate the performance of SCELM, we conduct extensive experiments on two microservice system datasets D1 and D2. Table 1 provides the detailed information of the datasets.

D1 is from a large-scale microservice system operated by an e-commerce company serving over 100 million users, with thousands of software changes weekly. The company has faced significant economic losses due to failed changes, emphasizing the need for effective change management. We studied 263 change cases over two years, selecting 54 representative cases for the experiment—40 erroneous changes and 14 normal changes. Failures in D1 include change issues related to container hardware, network, CPU, memory, node disk, and business-related failures. We used data from one week before the change as training data and from the week after the change as testing data. Due to confidentiality agreements, this dataset is not publicly available.

D2 is from a popular [12, 22, 37] microservice benchmark system: Hipster Shop [29], which allows us to inject various types of software changes (e.g., adding dead loops, modifying network configurations, introducing random delays in SQL queries) into microservices. We used Prometheus to collect time-series KPI data, including CPU usage, memory utilization, network flow, service success rate, transaction count, and response time. Two authors independently labeled KPI segments as anomalous or not, resolving any disagreements through discussion.

5.1.2 Baseline Approaches. We selected the current popular automated change methods (i.e., Lumos[24], FUNNEL[40], Gandalf[16], SCWarn[43], Kontrast[31]) as our baseline methods, which are mainly applicable to ECD. ChangeRCA[38] requires the use of service dependency graphs for service-level RCCA. Since the causal assumptions and data requirements adopted by ChangeRCA are not consistent with those of D1 and D2, we use the representative method PDiagnose[9] for RCCA. Since FT is not involved in the automated change approach, we adopt MicroCBR[17], a commonly used FT automation method in microservices, as our baseline in the FT phase. We use the parameters as specified in the respective work. For dataset-specific settings (e.g., window length), we adjust them based on the ranges provided or according to our data. Additionally, due to the absence of an ECD module in some methods and to maintain independent performance evaluation for each task, we assume known timestamps of failures in FT and RCCA evaluations.

5.1.3 Evaluation Metrics. Both ECD and FT are classification tasks. The former is a binary classification of whether a failure occurs, while the latter is a multi-classification problem of which type the current failure belongs to. During evaluations, we adopt True Positive (TP), False Positive (FP), and False Negative (FN), and then calculate: precision = $\frac{\text{TP}}{\text{TP+FP}}$, recall = $\frac{\text{TP}}{\text{TP+FN}}$, and $F1$-score = $2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision+recall}}$. We use the Weighted Average F1-score [3] for FT considering the imbalanced failure types. For RCCA, we introduce TopK as: TopK = $\frac{1}{N}\sum_{i=1}^{N}(g_i \in P_{i,1:K})$, to calculate the probability of the root cause within the top-K predicted candidates $P_{i,[1:K]}$, where g_i is the groundtruth root cause for the i-th failure case, and N is the number of failures for evaluation. Finally, AVG@5 = $\frac{1}{5}\sum_{K=1}^{5}$ TopK.

5.1.4 Implementations and Parameters. We use Python 3.9.19, Pytorch 2.3.0, scikit-learn 1.5.1, and langchain 0.2.10 to implement SCELM and baseline. We run experiments on a server with CUDA Version: 12.4 NVIDIA RTX A6000 (GPU). We use the flagship models qwen2[36], llama3[5], gemma2[30] open sourced by various companies as hyperparameters. We use Ollama (version is 0.1.48) for local deployment. We repeat each experiment five times and average the results to minimize the impact of randomness.

5.2 RQ1: How Does SCELM Perform in ECD, FT, and RCCA?

Table 2 presents a comparison of SCELM with baselines. SCELM consistently outperforms the baselines across all evaluation metrics, demonstrating superior performance in ECD, FT, and RCCA.

In the ECD phase, SCELM achieved F1-scores of 1.0 and 0.9421 on D1 and D2, respectively. In comparison, the baseline methods—SCWarn, Kontrast, Lumos, Funnel, and Gandalf—achieved average F1-scores of 0.8935, 0.888, 0.868, 0.869, and 0.8685, respectively. SCELM's average precision and recall were 1.000 and 0.946, respectively. This indicates that SCELM is highly effective at detecting erroneous changes, with minimal false negatives (which could lead to missed failure alerts and degraded service quality) and few false positives (minimizing wasted diagnostic efforts and preventing the blockage of legitimate changes).

We analyzed the performance gaps between SCELM and the baseline methods. Among the baselines, SCWarn achieved the best performance, with an average F1-score close to 0.9, benefiting from LSTM's strength in processing time series data. However, it fails to account for the semantic information in logs arising from changes. Kontrast, which uses contrastive learning to compare time series changes before and after a change, struggles to capture correlations between metrics and logs during failures. Lumos, relying on statistical tests, is effective only when significant data changes occur. Funnel, using iSST (improved Singular Spectrum Transform) for change point detection, is limited in its applicability across all types of time series data. Gandalf, based on Holt-Winters for anomaly detection, is restricted to seasonal KPIs.

In the FT and RCCA phases, SCELM achieved F1-scores of 0.964 and 0.865, and Top1 scores of 0.775 and 0.879, respectively, across both datasets. In contrast, MicroCBR only achieved F1-scores of 0.461 and 0.414, while PDiagnose had an AVG@5 of 0.150 and 0.507, which is worse than SCELM's Top1 performance. This can be attributed to MicroCBR and PDiagnose's limited sensitivity to fine-grained changes in the change process, as they fail to capture correlations between metrics and logs. Additionally, in PDiagnose, the analysis outcomes of each modality are influenced by the preceding modality, creating potential cascading effects.

Table 3 compares the efficiency of SCELM with the baseline methods. SCELM outperforms the baselines by completing more tasks per change case in the same time frame. SCELM's approach mirrors the workflow of real-world operations engineers, ensuring high performance with a lightweight design. Despite being slightly slower overall (e.g., SCWarn took 4.375 seconds for D1, while SCELM took

Table 2: Performances of ECD, FT, and RCCA. A dash ("-") indicates that the method does not address the respective problem.

Method	D1									D2								
	ECD			FT			RCCA			ECD			FT			RCCA		
	Precision	Recall	F1-score	Precision	Recall	F1-score	Top1	Top3	AVG@5	Precision	Recall	F1-score	Precision	Recall	F1-score	Top1	Top3	AVG@5
SCELM	1.000	1.000	1.000	0.964	0.964	0.964	0.775	0.900	0.975	1.000	0.891	0.942	0.861	0.870	0.865	0.879	0.932	0.932
SCWarn	0.900	0.857	0.878	-	-	-	-	-	-	0.929	0.891	0.909	-	-	-	-	-	-
Kontrast	0.925	0.860	0.892	-	-	-	-	-	-	0.891	0.876	0.884	-	-	-	-	-	-
Lumos	0.926	0.841	0.881	-	-	-	-	-	-	0.874	0.837	0.855	-	-	-	-	-	-
Funnel	0.875	0.875	0.875	-	-	-	-	-	-	0.858	0.867	0.863	-	-	-	-	-	-
Gandalf	0.900	0.818	0.857	-	-	-	-	-	-	0.880	0.880	0.880	-	-	-	-	-	-
MicroCBR	-	-	-	0.461	0.461	0.461	-	-	-	-	-	-	0.418	0.420	0.414	-	-	-
PDiagnose	-	-	-	-	-	-	0.025	0.100	0.150	-	-	-	-	-	-	0.067	0.307	0.507

Table 3: Processing Time for Each Change Case (in seconds).

Method	ECD	FT	RCCA	D1	D2
SCELM	✓	✓	✓	6.357	6.977
SCWarn	✓			4.375	2.189
Kontrast	✓			6.331	2.829
Lumos	✓			4.375	1.928
Funnel	✓			8.875	3.928
Gandalf	✓			3.605	2.920
MicroCBR		✓		47.937	19.279
PDiagnose			✓	0.357	0.179

Table 4: The Evaluation Results of Ablation Study.

Stage	Evaluation	D1			D2		
		SCELM	A1	A2	SCELM	A1	A2
ECD	Precision	1.000	1.000	1.000	1.000	0.764	1.000
	Recall	1.000	1.000	1.000	0.891	0.979	0.943
	F1-score	1.000	1.000	1.000	0.942	0.858	0.971
FT	Precision	0.964	0.864	0.864	0.870	0.825	0.838
	Recall	0.964	0.926	0.929	0.865	0.648	0.690
	F1-score	0.964	0.895	0.895	0.861	0.659	0.723
RCCA	Top1	0.775	0.000	0.100	0.879	0.147	0.542
	Top3	0.900	0.000	0.120	0.932	0.158	0.542
	AVG@5	0.975	0.000	0.120	0.932	0.163	0.542

6.357 seconds), SCELM completed three tasks (ECD, FT, RCCA) in that time, whereas SCWarn only handled ECD. This demonstrates SCELM's efficiency and suitability for end-to-end change processes.

Overall, the results highlight the practical applicability of SCELM, showcasing its ability to efficiently and effectively perform real-time ECD, FT, and RCCA in a unified framework.

5.3 RQ2: Ablation Study

To demonstrate the effectiveness of two key components of SCELM —(1) the natural language description of the data in the domain text, and (2) the detection algorithm for multimodal data—we conducted an ablation study. We created two variants of SCELM: A1 removes the data description in the domain text; A2 does not perform detection algorithm detection on multimodal data. Table 4 shows the results of each variant.

Effect of ECD. The experimental results indicate that removing both key components (natural language description and detection algorithm) has little effect on the ECD task. This can be explained by the fact that ECD is fundamentally anomaly detection, and LLMs are inherently able to recognize data fluctuations. In A2, where no detection algorithm is used, the model still performs satisfactorily with just the natural language description. This suggests that the physical meaning of the data, which is explicitly described in natural language, holds significant value. In real-world scenarios, engineers often make judgments based on the physical meaning of data, and the detection algorithm serves more as an auxiliary tool.

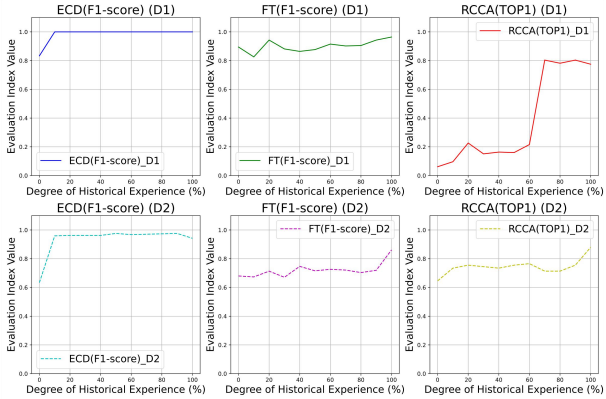

Figure 10: Experience Level Comparison Results.

Effect of FT. The impact on FT is similar to ECD. However, overall, SCELM outperforms both variants. This is because SCELM performs failure triage mainly based on the physical meaning of the data, which is consistent with how engineers perform triage in real-world situations. Engineers typically classify failures based on their semantic understanding—rooted in the physical meaning of the data—and LLMs excel at processing this type of semantic information. Hence, SCELM's performance remains superior even when certain components are removed.

Effect of RCCA. For RCCA, the removal of the natural language description significantly weakens SCELM's ability to identify the root cause. In the A1 variant, SCELM's root cause detection is nearly non-existent, with Top 5 results showing all empty values, especially in D1. This indicates that SCELMstruggles to pinpoint the root cause without the natural language context that explains the physical meaning of the data. When the detection algorithm is also removed (as in A2), the results are not as poor as A1, but they are still suboptimal. This suggests that both the detection algorithm and the natural language description of the data complement each other in the RCCA task. In the real world, engineers typically use both the results from detection algorithms and their understanding of the physical meaning of data to identify the root cause. Therefore, the combined effect of the detection algorithm and the semantic description in SCELMallows it to replicate this process and make more accurate root cause assessments.

5.4 RQ3: Hyperparameters Sensitivity

We use the degree of professional knowledge in RAG as a hyperparameter in SCELM to explore how historical experience impacts algorithm performance. This provides guidance for deploying the algorithm in real-world scenarios. We randomly extract historical experiences from two datasets, D1 and D2, at different proportions.

Table 5: Performance With and Without RAG.

Dataset	avg_reason	avg_reason (no RAG)	avg_solution	avg_solution (no RAG)
D1	0.840	0.567	0.800	0.641
D2	0.968	0.778	0.944	0.632

The experimental results are shown in Figure 10, where the solid line represents D1 and the dotted line represents D2.

Cold Start (0% Historical Experience). With no historical experience (a cold start), SCELM still shows reasonable performance, particularly in ECD.

Increase in Historical Experience (Up to 10%). As the historical experience increases, performance steadily improves. Beyond 10% historical experience, further improvements are minimal. This could be because ECD is anomaly detection, which relies heavily on data fluctuations and the physical meaning of the data itself.

Impact on FT (Beyond 10%). When experience level reaches 20%, further improvements in FT become less noticeable, and some slight downward trends appear in the D1 dataset. This is due to the small number of cases in D1—when extracting 10%, only two cases are sampled, and they may either belong to the same category or different categories. Thus, in real-world deployment, even when dealing with a small number of cases, it is important to cover a wider range to help the algorithm learn more effectively.

Effect on RCCA (Beyond 70%). After 70% historical experience, the improvement in RCCA becomes negligible, particularly in D1. The large number of cases in D2 helps stabilize the results, while D1's smaller case count limits the model's ability to meet learning requirements until around 70% historical experience. In real-world deployment, when historical experience is very limited, the algorithm can focus on ECD and FT. Once the number of different cases exceeds 20, the improvement in RCCA is substantial.

5.5 RQ4: How RAG Mitigates Hallucinations and Improves Change Expertise?

To better utilize SCELM and reduce manual involvement in the change process, we studied the professionalism of the output text generated by SCELM and explored the effect of RAG on reducing hallucinations in LLMs. We used the failure causes and solutions from historical erroneous changes as the benchmark text and compared them to the failure causes and solutions generated by SCELM as the reference text. The cosine similarity between the benchmark text and the reference text was calculated [23].

As shown in Table 5, *avg_reason* represents the average cosine similarity of the failure causes, and *avg_solution* represents the average cosine similarity of the solutions. The table clearly shows that, whether for D1 or D2, there is a significant difference in the output text when SCELMis used with and without RAG. With RAG's participation, the similarity for the D1 dataset remains above 0.8, and in D2, it even exceeds 0.9.

To verify this, we specifically checked the model's output and found that, with RAG's participation, the output text did not exhibit the nonsensical patterns (hallucinations) typically caused by LLMs. While the similarity did not reach 100%, this is understandable due to the inherent differences between cases—failure causes and solutions vary across different situations. Thus, RAG effectively reduces the impact of hallucinations in LLMs and enhances the professionalism and accuracy of their output, especially in the context of change management.

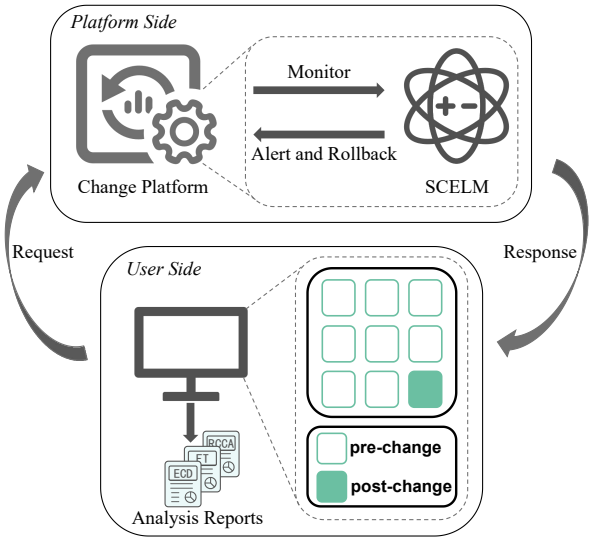

Figure 11: The Deployment of SCELM on The Change Platform.

5.6 RQ5: LLMs with Small Parameter Scales

Table 6 presents the experimental results of LLMs with different parameter scales. It is evident that the model with 7 billion parameters achieves the best performance. When the model has 2 billion parameters, its performance on the ECD and FT tasks remains satisfactory, but there is a notable degradation in performance during the RCCA phase. This decline can primarily be attributed to the 2-billion-parameter model's limited capacity to capture the semantic nuances of domain changes, likely due to the constraints imposed by its smaller parameter size. We have tried using LLMs with larger parameters, but due to our graphics card bottleneck, LLMs with larger parameters run very slowly, which is not enough to meet the timeliness requirements of changes. In theory, LLMs with larger parameters have a deeper understanding of semantics and the results may be more accurate, especially for RCCA. However, from the current results, LLMs with 7 billion parameters is sufficient and has a certain degree of economy with high cost performance.

6 Implementation

SCELM has been deployed in an automated change platform for a large SaaS microservice system. As illustrated in Figure 11, when a new change occurs, the user initiates it through the platform, which generates a change order and begins execution. SCELM monitors the change status in real-time, displays updates to the user, and issues alerts upon detecting erroneous changes. It triggers system rollbacks and provides a detailed change analysis report to aid engineers in corrective actions.

Deployment effectiveness. SCELM has been operational for over 11 months, monitoring thousands of changes weekly. Feedback from OCEs indicates that it detects nearly all erroneous changes, achieving over 95% F1-score in classification and 75% localization accuracy, aligning with evaluation results. Compared to traditional methods, SCELM reduces erroneous change resolution times by 90%, significantly enhancing accident-handling efficiency. OCEs report simplified, faster, and more automated change assessments,

Table 6: Performance of LLMs with Small Parameter Scales.

Stage	Evaluation	D1		D2	
		SCELM(7b)	SCELM(2b)	SCELM(7b)	SCELM(2b)
ECD	Precision	1.000	1.000	1.000	0.9641
	Recall	1.000	0.750	0.984	0.979
	F1-score	1.000	0.857	0.992	0.972
FT	Precision	0.964	0.864	0.870	0.514
	Recall	0.964	0.929	0.865	0.547
	F1-score	0.964	0.895	0.861	0.520
RCCA	Top1	0.775	0.022	0.879	0.214
	Top3	0.900	0.022	0.932	0.219
	AVG@5	0.975	0.022	0.932	0.219

highlighting SCELM's strong generalization capabilities and plans for broader application.

7 Discussion

7.1 Lessons Learned

One of the lessons we learned is the scarcity of data, especially during the cold start phase, which can lead to inaccurate results. To address this, practitioners should use change data early on to help the model build foundational knowledge. Organizing internal records and historical cases would provide better context.

Although this study focuses on RAG technology due to limited data, fine-tuning LLMs could improve performance in specific scenarios. Future work will combine fine-tuning with RAG to optimize performance across more scenarios. Additionally, transfer learning and data augmentation could also help mitigate the cold start issue by transferring data from similar scenarios or generating synthetic data, allowing the model to adapt quickly.

Another lesson we learned is the validity of change work order information, as companies may differ in how they record data. Engineers sometimes fail to report the true causes of failures. To improve the reliability of historical data, companies should ensure work order accuracy. This is essential for tracing past failures and guiding future change evaluations.

7.2 Threats to Validity

Threats to validity mainly arise from variations in the capabilities of large models. While SCELM uses large models for specialized tasks, their performance can vary depending on domain-specific features and anomaly detection. Even with the same SCELM framework, different training data, parameter settings, and model architectures can lead to different results.

To mitigate this, we recommend performance evaluations and multimodal integration strategies. Models should be chosen based on the specific context, and ensemble methods can be used to improve accuracy by combining outputs from different models. An adaptive evaluation system can adjust model weights based on performance, minimizing the impact of model differences.

These strategies will enhance the validity and robustness of SCELM in real-world applications, helping it adapt to evolving change scenarios and providing reliable support for decision-making.

8 Related Work

8.1 Software Changes

Software changes have been a popular research domain in academia and industry for several years [18, 35, 38, 42]. In order to improve the reliability of software changes, it is crucial to identify failure software changes in a timely manner and find out the root causes to solve them. Existing erroneous software change identification approaches [16, 40, 43], Lumos[24] majorly regard this problem as an anomaly detection task, utilizing anomaly detection (or change point detection [19–21]) algorithms to apply to this problem directly. For instance, multimodal LSTM in SCWarn[43], improved Singular Spectrum Transform (iSST) in Funnel[40], Holt-Winters in Gandalf[16], A/B Test in Lumos[24] and contrastive learning in Kontrast[31]. However, these approachs all focus on ECD. Although ChangeRCA[38] focuses on RCCA, the root cause analysis only stays at the microservice level and cannot locate which indicators or logs in specific services are the root causes. These change techniques only cover certain stages and do not span the entire change lifecycle. In our approach, we use multimodal data fusion combined with LLM to identify failure software changes and find out the root causes to provide targeted solutions and adjustment suggestions, covering the entire life cycle of software changes.

8.2 LLMs-RAG

LLMs have achieved remarkable success, but they continue to exhibit significant limitations, particularly in specialized domains or knowledge-intensive tasks. These limitations include a tendency to produce "hallucinations"—factually inaccurate or fabricated responses—when addressing queries that exceed their training data or require real-time information. To address these challenges, RAG enhances LLMs by leveraging external knowledge bases. Through semantic similarity calculations, RAG retrieves relevant document chunks, enabling LLMs to incorporate external information. This approach significantly mitigates the problem of generating factually incorrect content[6]. RAG technologies have been successfully applied across various domains, including fault diagnosis in power grids[11] and medical decision-making [10]. In our study, we applied RAG techniques to the domain of software change assessment management and observed encouraging outcomes.

9 Conclusion

Ensuring reliability during software changes is crucial. This paper proposes SCELM, an innovative unsupervised automated change assessment framework that integrates ECD, FT, and RCCA using multimodal techniques. SCELM fully automates the key steps in software change assessment by leveraging LLMs and their multi-task learning capabilities. While LLMs have been applied in various domains (fine-tuning and RAG), our research reveals that in the change management domain, existing data is insufficient, and fine-tuning methods show limited effectiveness. In contrast, RAG demonstrates strong expressiveness, and our approach achieves promising results across multiple change datasets. This work sets a benchmark for future research, provides a standard for LLM applications in change management, and enriches the evaluation metrics for LLMs.

References

[1] Betsy Beyer, Chris Jones, Jennifer Petoff, and Niall Richard Murphy. 2016. *Site reliability engineering: How Google runs production systems.* " O'Reilly Media, Inc.".

[2] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374* (2021).

[3] Nancy Chinchor and Beth M Sundheim. 1993. MUC-5 evaluation metrics. In *Fifth Message Understanding Conference (MUC-5): Proceedings of a Conference Held in Baltimore, Maryland, August 25-27, 1993*.

[4] Hanxing Ding, Liang Pang, Zihao Wei, Huawei Shen, and Xueqi Cheng. 2024. Retrieve only when it needs: Adaptive retrieval augmentation for hallucination mitigation in large language models. *arXiv preprint arXiv:2402.10612* (2024).

[5] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783* (2024).

[6] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997* (2023).

[7] Aman Gupta, Anup Shirgaonkar, Angels de Luis Balaguer, Bruno Silva, Daniel Holstein, Dawei Li, Jennifer Marsman, Leonardo O Nunes, Mahsa Rouzbahman, Morris Sharp, et al. 2024. RAG vs Fine-tuning: Pipelines, Tradeoffs, and a Case Study on Agriculture. *arXiv preprint arXiv:2401.08406* (2024).

[8] Pinjia He, Jieming Zhu, Zibin Zheng, and Michael R Lyu. 2017. Drain: An online log parsing approach with fixed depth tree. In *2017 IEEE international conference on web services (ICWS)*. IEEE, 33–40.

[9] Chuanjia Hou, Tong Jia, Yifan Wu, Ying Li, and Jing Han. 2021. Diagnosing performance issues in microservices with heterogeneous data source. In *2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*. IEEE, 493–500.

[10] Xinke Jiang, Yue Fang, Rihong Qiu, Haoyu Zhang, Yongxin Xu, Hao Chen, Wentao Zhang, Ruizhe Zhang, Yuchen Fang, Xu Chu, et al. 2024. TC-RAG: Turing-Complete RAG's Case study on Medical LLM Systems. *arXiv preprint arXiv:2408.09199* (2024).

[11] Liu Jing and Amirul Rahman. 2024. Fault Diagnosis in Power Grids with Large Language Model. *arXiv preprint arXiv:2407.08836* (2024).

[12] Lars Larsson, William Tärneberg, Cristian Klein, Maria Kihl, and Erik Elmroth. 2021. Adaptive and application-agnostic caching in service meshes for resilient cloud applications. In *2021 IEEE 7th International Conference on Network Softwarization (NetSoft)*. IEEE, 176–180.

[13] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems* 33 (2020), 9459–9474.

[14] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems* 33 (2020), 9459–9474.

[15] Huayang Li, Yixuan Su, Deng Cai, Yan Wang, and Lemao Liu. 2022. A survey on retrieval-augmented text generation. *arXiv preprint arXiv:2202.01110* (2022).

[16] Ze Li, Qian Cheng, Ken Hsieh, Yingnong Dang, Peng Huang, Pankaj Singh, Xinsheng Yang, Qingwei Lin, Youjiang Wu, Sebastien Levy, et al. 2020. Gandalf: An intelligent,{End-To-End} analytics service for safe deployment in {Large-Scale} cloud infrastructure. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. 389–402.

[17] Fengrui Liu, Yang Wang, Zhenyu Li, Rui Ren, Hongtao Guan, Xian Yu, Xiaofan Chen, and Gaogang Xie. 2022. MicroCBR: Case-Based Reasoning on Spatio-temporal Fault Knowledge Graph for Microservices Troubleshooting. In *International Conference on Case-Based Reasoning*. Springer, 224–239.

[18] Ajay Mahimkar, Carlos Eduardo de Andrade, Rakesh Sinha, and Girotharan Rana. 2021. A composition framework for change management. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*. 788–806.

[19] Ajay Mahimkar, Zihui Ge, Jia Wang, Jennifer Yates, Yin Zhang, Joanne Emmons, Brian Huntley, and Mark Stockert. 2011. Rapid detection of maintenance induced changes in service performance. In *Proceedings of the Seventh COnference on Emerging Networking EXperiments and Technologies*. 1–12.

[20] Ajay Mahimkar, Zihui Ge, Jennifer Yates, Chris Hristov, Vincent Cordaro, Shane Smith, Jing Xu, and Mark Stockert. 2013. Robust assessment of changes in cellular networks. In *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies*. 175–186.

[21] Ajay Anil Mahimkar, Han Hee Song, Zihui Ge, Aman Shaikh, Jia Wang, Jennifer Yates, Yin Zhang, and Joanne Emmons. 2010. Detecting the performance impact of upgrades in large operational networks. In *Proceedings of the ACM SIGCOMM 2010 Conference*. 303–314.

[22] John Paul Martin, A Kandasamy, and K Chandrasekaran. 2020. CREW: Cost and Reliability aware Eagle-Whale optimiser for service placement in Fog. *Software: Practice and Experience* 50, 12 (2020), 2337–2360.

[23] Anirudh Phukan, Harshit Kumar Morj, Apoorv Saxena, Koustava Goswami, et al. 2024. Beyond Logit Lens: Contextual Embeddings for Robust Hallucination Detection & Grounding in VLMs. *arXiv preprint arXiv:2411.19187* (2024).

[24] Jamie Pool, Ebrahim Beyrami, Vishak Gopal, Ashkan Aazami, Jayant Gupchup, Jeff Rowland, Binlong Li, Pritesh Kanani, Ross Cutler, and Johannes Gehrke. 2020. Lumos: A library for diagnosing metric regressions in web-scale applications. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2562–2570.

[25] Alec Radford. 2018. Improving language understanding by generative pre-training. (2018).

[26] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.

[27] Andy Singleton. 2016. The economics of microservices. *IEEE Cloud Computing* 3, 5 (2016), 16–20.

[28] Shamane Siriwardhana, Rivindu Weerasekera, Elliott Wen, Tharindu Kaluarachchi, Rajib Rana, and Suranga Nanayakkara. 2023. Improving the domain adaptation of retrieval augmented generation (RAG) models for open domain question answering. *Transactions of the Association for Computational Linguistics* 11 (2023), 1–17.

[29] Clay Smith. 2022. hipster-shop. //https://github.com/lightstep/hipster-shop.

[30] Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118* (2024).

[31] Xuanrun Wang, Kanglin Yin, Qianyu Ouyang, Xidao Wen, Shenglin Zhang, Wenchi Zhang, Li Cao, Jiuxue Han, Xing Jin, and Dan Pei. 2022. Identifying erroneous software changes through self-supervised contrastive learning on time series data. In *2022 IEEE 33rd International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 366–377.

[32] Wikipedia contributors. 2024. 2021 Facebook outage — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=2021_Facebook_outage&oldid=1260851067. [Online; accessed 11-January-2025].

[33] Canhua Wu, Nengwen Zhao, Lixin Wang, Xiaoqin Yang, Shining Li, Ming Zhang, Xing Jin, Xidao Wen, Xiaohui Nie, Wenchi Zhang, et al. 2021. Identifying root-cause metrics for incident diagnosis in online service systems. In *2021 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 91–102.

[34] Yifan Wu, Bingxu Chai, Ying Li, Bingchang Liu, Jianguo Li, Yong Yang, and Wei Jiang. 2023. An empirical study on change-induced incidents of online service systems. In *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 234–245.

[35] Yong Xu, Xu Zhang, Chuan Luo, Si Qin, Rohit Pandey, Chao Du, Qingwei Lin, Yingnong Dang, and Andrew Zhou. 2021. CARE: Infusing causal aware thinking to root cause analysis in cloud system. In *Proceedings of the 1st Workshop on High Availability and Observability of Cloud Systems*. 1–3.

[36] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2. 5 Technical Report. *arXiv preprint arXiv:2412.15115* (2024).

[37] Guangba Yu, Pengfei Chen, Hongyang Chen, Zijie Guan, Zicheng Huang, Linxiao Jing, Tianjun Weng, Xinmeng Sun, and Xiaoyun Li. 2021. Microrank: End-to-end latency issue localization with extended spectrum analysis in microservice environments. In *Proceedings of the Web Conference 2021*. 3087–3098.

[38] Guangba Yu, Pengfei Chen, Zilong He, Qiuyu Yan, Yu Luo, Fangyuan Li, and Zibin Zheng. 2024. ChangeRCA: Finding Root Causes from Software Changes in Large Online Systems. *Proceedings of the ACM on Software Engineering* 1, FSE (2024), 24–46.

[39] Shenglin Zhang, Pengxiang Jin, Zihan Lin, Yongqian Sun, Bicheng Zhang, Sibo Xia, Zhengdan Li, Zhenyu Zhong, Minghua Ma, Wa Jin, et al. 2023. Robust failure diagnosis of microservice system through multimodal data. *IEEE Transactions on Services Computing* 16, 6 (2023), 3851–3864.

[40] Shenglin Zhang, Ying Liu, Dan Pei, Yu Chen, Xianping Qu, Shimin Tao, and Zhi Zang. 2015. Rapid and robust impact assessment of software changes in large internet-based services. In *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*. 1–13.

[41] Tianjun Zhang, Shishir G Patil, Naman Jain, Sheng Shen, Matei Zaharia, Ion Stoica, and Joseph E Gonzalez. 2024. Raft: Adapting language model to domain specific rag. *arXiv preprint arXiv:2403.10131* (2024).

[42] Xu Zhang, Chao Du, Yifan Li, Yong Xu, Hongyu Zhang, Si Qin, Ze Li, Qingwei Lin, Yingnong Dang, Andrew Zhou, et al. 2021. Halo: Hierarchy-aware fault localization for cloud systems. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 3948–3958.

[43] Nengwen Zhao, Junjie Chen, Zhaoyang Yu, Honglin Wang, Jiesong Li, Bin Qiu, Hongyu Xu, Wenchi Zhang, Kaixin Sui, and Dan Pei. 2021. Identifying bad software changes via multimodal anomaly detection for online service systems. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 527–539.