MERA: A Multi-Expert LLM Framework for Severity Assessment of User Tickets in Mobile OS Beta Testing

Yongqian Sun[1], Yimin Zuo[1,2,*], Qingliang Zhang[1], Bowen Hao[1], Wenwei Gu[1,†]
Jiaqi Luan[1], Sixu Zhou[1], Shenglin Zhang[1], Dan Pei[3]
[1] Nankai University, China [2] Huawei Inc., China [3] Tsinghua University, China
{sunyongqian,wwgu,zhangsl}@nankai.edu.cn
{2213023,zhangqingliang,haobw,1120250451,2320230843}@mail.nankai.edu.cn
zuoyimin@h-partners.com peidan@tsinghua.edu.cn

Abstract

Large-scale mobile OS beta testing generates massive ticket volumes, creating a triage bottleneck in which assigning severity dictates engineering priority. Automating this is difficult: unlike server logs with rich telemetry, beta tickets rely on vague, symptom-driven user descriptions, which confound standard AIOps tools. Traditional classifiers based on deep learning or pre-trained language models (PLMs) further struggle to adapt to the rapid concept drift triggered by frequent feature updates, and their black-box nature fails to provide the reasoning transparency that engineers require for trust. We observe that experienced triage engineers resolve these challenges by triangulating across user intent, system context, and historical precedent—a discipline that no single monolithic model captures. We present *MERA* (**M**ulti-expert **E**vidence **R**easoning and **A**rbitration), the first production-deployed multi-expert LLM framework for severity assessment in telemetry-void mobile OS beta testing. *MERA* operationalizes this discipline by decoupling severity assessment into three expert perspectives and fusing their evidence into an auditable decision. To meet strict online latency budgets, *MERA* distills the reasoning capability of a large teacher model into three lightweight experts, and a dynamic fusion layer weighs their evidence to produce a severity decision with human-reviewable rationales. In Huawei's HarmonyOS beta testing, *MERA* serves as the default severity grader for all incoming beta tickets, with engineers performing spot-checks. Over five months it processed more than 370,000 tickets at 85% accuracy, substantially outperforming baselines under concept drift and reducing manual triage effort by 60%.

CCS Concepts

• **Software and its engineering** → *Software defect analysis.*

Keywords

Severity Assessment, Large Language Models, AIOps, Multi-Expert Reasoning

1 Introduction

With the growth of the smartphone ecosystem, mobile operating systems have become increasingly complex and evolve at a fast cadence [1, 2]. Defects are therefore inevitable, spanning minor user experience (UX) glitches (*e.g.*, intermittent UI stutters) to high-impact failures (*e.g.*, core features becoming unusable on certain devices/builds), and even severe security vulnerabilities [3, 4]. Such defects not only degrade user experience but also carry business and reputational costs [5, 6].

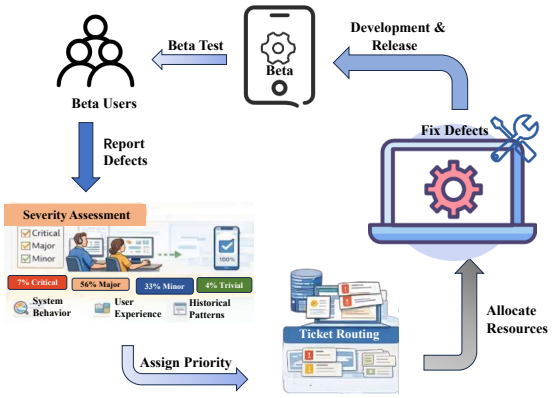


Figure 1: Workflow of beta testing.

To surface and fix defects before release, mobile OS providers rely on large-scale beta testing, where users report defects from pre-release builds as tickets [7]. As shown in Figure 1, these tickets enter a production triage pipeline in which *severity assessment* (Trivial, Minor, Major, Critical) determines routing, prioritization, and fix scheduling [8]. In our deployment at Huawei, this stage handles over 5,000 tickets per day under strict online latency requirements, and misgrading directly translates into wasted engineering effort or delayed handling of high-impact defects. Figure 2 shows two real beta tickets with similar surface forms but markedly different severity.

Assigning calibrated severity labels at this scale is non-trivial. Tickets are free-form, underspecified, and context-dependent [9, 10], and engineers must reconcile conflicting cues (*e.g.*, emotional user language vs. technical impact) largely through tacit, experience-driven intuition. This makes manual triage inconsistent and establishes severity assessment as a persistent operational bottleneck [11]. Existing automated approaches, however, do not transfer well to this setting. Metric-driven AIOps methods assume rich runtime telemetry that beta tickets simply lack; supervised classifiers built on PLMs [12] degrade quickly under release-driven concept drift and expose little auditable rationale; and monolithic LLMs, despite stronger semantic reasoning [13], tend to entangle subjective user signals with objective technical evidence within a single reasoning chain. Compounding these modeling gaps, a strict online latency budget rules out large-parameter models and multi-stage inference pipelines. We formalize this setting and analyze these gaps in Section 2.1 and Section 2.2.

* Work done during an internship at Huawei. † Corresponding author.

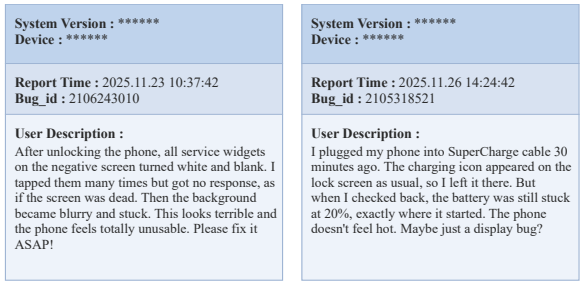


Figure 2: Example of tickets.

We therefore propose *MERA* (**M**ulti-expert **E**vidence **R**easoning and **A**rbitration), a production-oriented framework for severity assessment in mobile OS beta testing. At the paradigm level, *MERA* grounds severity judgment in the semantic reasoning capability of LLMs rather than in telemetry or surface keywords, producing rationales that engineers can audit and offering greater resilience to release-driven drift than pattern-matching classifiers. Within this paradigm, *MERA* introduces three architectural components:

- **Multi-expert decomposition.** To prevent conflicting signals from collapsing into a single chain of thought, *MERA* assigns grading to three specialized experts—*User Experience*, *System Risk*, and *Historical Consistency*—grounded in Huawei's internal Production Severity Rubric (Section 2.2.4).
- **Ticket-anchored arbitration.** To adapt expert weighting to each ticket rather than relying on fixed rules such as majority voting, *MERA* employs a ticket-anchored attention mechanism that dynamically weighs expert evidence based on each ticket's semantic context.
- **Teacher–student distillation.** To deploy multi-expert reasoning under a 1.8 s per-ticket budget, *MERA* moves expensive reasoning offline through teacher–student distillation, compressing traces from a high-capacity teacher into lightweight experts served online. This brings per-ticket latency from 2.07 s—which violates the production budget—down to 1.48 s, a 28.5% reduction, while preserving teacher-scale reasoning quality.

On a manually verified golden set, *MERA* achieves 91.2% accuracy; in five months of production deployment at Huawei, it has processed over 370,000 tickets at 85% accuracy and reduced manual triage labor by approximately 60%.

In summary, the main contributions of this paper are:

- We characterize severity assessment in mobile OS beta testing as a production-critical automation problem with four defining properties: telemetry-void reports, release-driven concept drift, the need for auditable decisions, and strict online latency budgets. We further show that neither classifier-based pipelines nor monolithic LLM solutions adequately address this combination of constraints.
- We present *MERA*, the first production-deployed multi-expert LLM framework for severity assessment in mobile OS beta testing. *MERA* combines three architectural components—multi-expert decomposition, ticket-anchored arbitration, and teacher–student distillation—on top of an LLM-based semantic reasoning backbone.
- We evaluate and deploy *MERA* in a real mobile OS beta pipeline. *MERA* reaches an $F1_{macro}$ of 0.906 offline, outperforming the strongest baseline by over 5 percentage points, and has sustained 85% accuracy over 370,000 production tickets in five months.

2 Background and Motivation

2.1 Problem Formulation

Let $T^{(t)} = \{T_1^{(t)}, \ldots, T_{N_t}^{(t)}\}$ denote the stream of beta tickets associated with release t. Each ticket is represented as $T_i^{(t)} = (W_i^{(t)}, M_i^{(t)})$, where $W_i^{(t)}$ is the user's free-form textual description and $M_i^{(t)}$ denotes available metadata (e.g., device model, build version).

The task is to predict a severity label $y_i^{(t)} \in \mathcal{Y}$, where $\mathcal{Y}$ is an ordinal label set:

$$\mathcal{Y} = \{\textsc{Trivial}, \textsc{Minor}, \textsc{Major}, \textsc{Critical}\}. \quad (1)$$

In deployment, the assessment function is expected to produce both a severity prediction and a supporting rationale:

$$(\hat{y}_i^{(t)}, r_i^{(t)}) = F\big(T_i^{(t)}, K\big), \quad (2)$$

where K denotes optional organizational knowledge, $r_i^{(t)}$ is the rationale, and F must operate under a strict online latency budget $\text{Latency}(F) \le \tau$, with $\tau = 1.8$ s per ticket in our deployment.

This formulation differs from server-side incident triage in three ways that shape the rest of this paper. First, beta tickets carry no runtime telemetry—no logs, traces, or metrics—so F must decide from $W_i^{(t)}$ and sparse metadata alone. Second, the label space $\mathcal{Y}$ is ordinal, so prediction errors are not equally costly and must be evaluated accordingly. Third, the release index t is not a passive superscript: the underlying distribution $p(W, M, y \mid t)$ shifts non-trivially across releases, making single-snapshot supervision insufficient. Together with the latency budget and the rationale requirement, these properties define severity assessment in mobile OS beta testing as a rationale-producing ordinal decision problem under telemetry absence, release-driven drift, and real-time constraints—the setting we analyze in Section 2.2.

2.2 Motivation

Severity misgrading translates directly into operational cost: an incorrectly escalated ticket consumes 15–30 engineer-hours through emergency triage and cross-team coordination, while under-grading a critical defect can trigger incident handling exceeding 100 engineer-hours. Existing tools fall short of industrial triage demands in three distinct ways, which we elaborate below.

2.2.1 *Challenge 1: Telemetry Void and Auditability Gap.*

Existing Practice. In production beta triage, engineers assess severity by reading free-form user descriptions alongside sparse context such as device models and build versions [7]. This step is heavily symptom-driven and manual.

The Industrial Gap. Two issues compound. First, tickets reveal user-visible symptoms but lack the runtime logs or traces needed to verify technical severity—we refer to this as the *telemetry void*—so prior AIOps methods [14, 15] that assume rich system metrics cannot transfer. Second, conventional text classifiers [12, 16, 17] expose no auditable rationale, leaving severity decisions unsubstantiated and difficult to trust during post-incident root-cause analysis. Closing this gap therefore requires reasoning that can extract meaning

from sparse symptomatic text and expose the basis of each decision for engineer review.

2.2.2 *Challenge 2: Release-Driven Drift.*

Existing Practice. Industrial triage systems typically build automated severity assessment on historical tickets—defensible when the operational environment is stable. In mobile OS beta testing, however, tickets originate from continuously evolving releases and diverse device–build combinations [18].

The Industrial Gap. This creates persistent *release-driven drift*: new functionalities introduce issue types with no close counterpart in past tickets, and the same surface description may imply different technical impact under different versions or device combinations. Models that rely on historical label patterns degrade quickly on newly emerging failures—an unsustainable trajectory for a system that must ship with every release. Sustained accuracy therefore demands reasoning over the semantics of each report rather than fitting to past label patterns alone.

2.2.3 *Challenge 3: Perspective Entanglement in Monolithic LLMs.*

Existing Practice. The two preceding challenges point toward LLMs, whose semantic reasoning promises both auditable rationales (Challenge 1) and generalization under drift (Challenge 2). Our initial attempts therefore deployed a centralized strategy: a high-capacity model processes raw ticket text and emits a severity label in a single inference pass.

The Industrial Gap. Pilot evaluations revealed a systematic failure we call *Perspective Entanglement*. A single unified reasoning process is easily skewed by "emotional noise"—a user's frustrated tone or exaggerated description that overshadows the underlying technical indicators. A minor UI glitch with intense complaints was often escalated to Major. This matches findings in structured decision-making research: forcing a single model to reconcile competing evidence dimensions (subjective user impact vs. objective system risk) within one execution trace causes reasoning fidelity to fluctuate [19, 20]. A monolithic reasoning path is therefore insufficient on its own—the LLM backbone must be paired with structural decomposition that isolates competing evidence dimensions.

2.2.4 *Design Requirements.*

DR1: Multi-Expert Decomposition. Challenges 1 and 2 together favor LLM-based semantic reasoning over classical telemetry-driven or pattern-matching approaches, while Challenge 3 shows that LLM reasoning alone is insufficient: a single reasoning chain must be structurally decomposed on top of the LLM backbone. The remaining question is along which dimensions decomposition should occur.

We ground this choice in an institutional artifact already in use: Huawei's internal *Production Severity Rubric*, the formal grading standard that triage engineers are required to follow, whose four weighted dimensions map to the severity bands of Section 2.1. As a codified, organization-wide standard, it provides a stronger basis for expert design than informal observation. As Table 1 shows, the four dimensions cluster into three perspectives by the type of evidence each requires—user-facing impact, system-level state, and cross-time grading consistency—which we instantiate as MERA's three experts: **User Experience (UX)**, **System Risk (SS)**, and **Historical Consistency (HC)**, each configured with the knowledge its perspective demands (no retrieval for UX, technical specifications C_{spec} for SS, and the historical ticket base C_{hist} for HC).

Table 1: Mapping from the four-dimension Production Severity Rubric to MERA's three experts.

Rubric Dimension (Weight)	Perspective	Expert
Function/Scenario Category – Device View (18%)	User-facing impact	UX
Fault Impact – UX subitems (part of 35%)	User-facing impact	UX
Fault Impact – functional/safety/stability (part of 35%)	System-level state	SS
Fault Recovery Condition (15%)	System-level state	SS
Historical Grading Consistency (32%)	Cross-time consistency	HC

This decomposition also aligns with a broader direction in the LLM literature, where decomposing complex decisions along complementary evidence axes has been shown to improve reasoning reliability over monolithic chains [19–22]. Other plausible dimensions (reproducibility, business impact) are either absorbed into existing rubric subitems or, in the telemetry-void beta setting, not reliably observable from ticket text. *MERA* accordingly decomposes severity reasoning along these three axes, so that each kind of evidence is reasoned about on its own terms before fusion.

DR2: Ticket-Anchored Arbitration. Decomposing the decision into three experts solves one problem but creates another: the experts will disagree, and *which* expert should dominate depends on the ticket at hand. A crash report is largely a System-Risk question; an ambiguous UX complaint on a long-standing feature is primarily a Historical-Consistency question. Static fusion rules such as majority voting cannot capture this context dependence. *MERA* therefore needs a fusion mechanism that conditions on each ticket's semantic content and dynamically adjusts how much weight each expert carries—an arbitration module anchored to the ticket itself rather than to fixed coefficients.

DR3: Teacher–Student Distillation. DR1 and DR2 together define an architecture that performs high-quality reasoning but is not, by default, cheap enough to serve in production. Running three high-capacity experts per ticket easily exceeds the 1.8 s latency required for online triage. Rather than compromise on reasoning depth, *MERA* shifts the computational burden offline through teacher–student distillation: expensive reasoning traces generated by a large teacher model are distilled into a lightweight student backbone that serves all three experts online. This preserves the multi-expert design of DR1–DR2 while meeting the production latency constraint.

3 Methodology

We present *MERA*, a production-oriented framework for severity assessment in mobile OS beta testing. As illustrated in Figure 3, *MERA* operates in two stages: (1) an **offline stage** that distills three specialized student experts (DR1, DR3), trains a ticket-anchored decision module (DR2), and indexes organizational knowledge into two retrieval corpora; and (2) an **online stage** that executes the three experts sequentially on a shared backbone with dynamic LoRA adapter switching, then invokes the decision module to produce the final severity label.

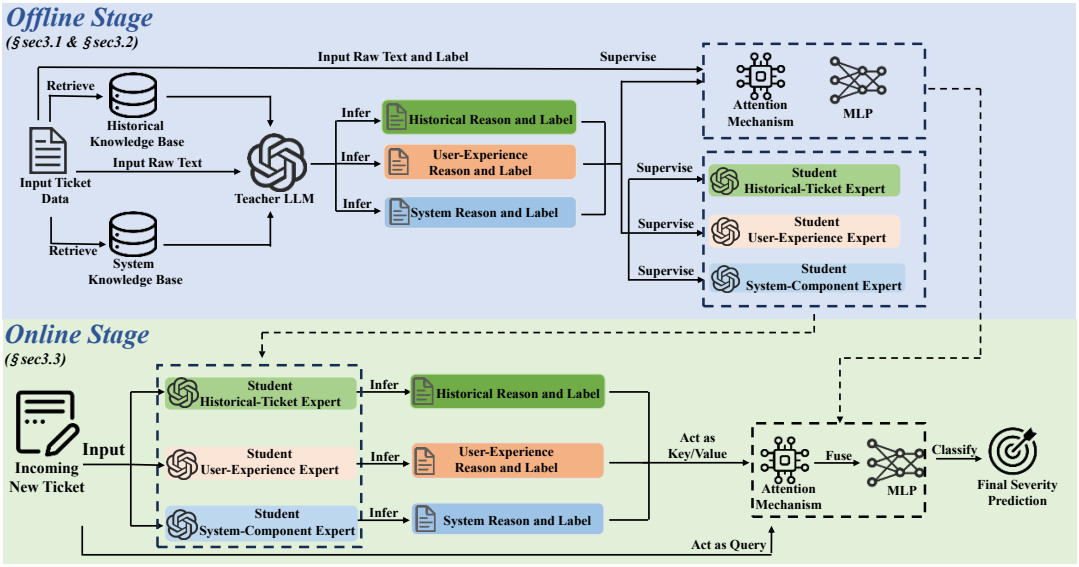


Figure 3: The framework of *MERA*.

3.1 Offline Expert Construction

This phase realizes DR1 through three LoRA adapters specialized for UX, SS, and HC reasoning, and DR3 by compressing high-capacity teacher reasoning traces into a shared lightweight backbone, so that all three experts together fit within the 1.8 s per-ticket latency budget.

3.1.1 Knowledge Grounding and Corpus Indexing. We construct two heterogeneous retrieval corpora that jointly instantiate the organizational knowledge $K = C_{hist} \cup C_{spec}$ introduced in Section 2.1:

- **Historical Ticket Base (C_{hist}):** Labeled tickets from the training split are embedded into a dense vector space [23]. Each entry serves as a few-shot precedent to ground the **Historical Consistency (HC)** expert in validated triage patterns.
- **Technical Specification Base (C_{spec}):** Internal technical documents are parsed into component-centric nodes using `Qwen3-VL-8B-Instruct`, chunked, and stored in a vector database. These nodes supply the **System Risk (SS)** expert with hardware–software dependencies retrieved by the incoming ticket's semantic signature.

3.1.2 Reasoning Distillation and Role Adaptation. **Trace Generation and Correctness Filtering.** We utilize `DeepSeek-R1-32B` [24] as a teacher model to generate structured reasoning traces for three independent roles: **UX**, **SS**, and **HC**. Traces are elicited via the **RGCIE** prompt design principle [25, 26], which structures each expert's prompt into five components: *Role*, *Goal*, *Constraints*, *Instructions*, and *Example*. As Figure 4 shows, the three experts share an identical *Constraints* slot—a unified JSON output contract that enables downstream fusion—but are differentiated primarily through the *Example* slot: SS retrieves from C_{spec}, HC from C_{hist}, and UX uses no retrieval.

To retain only valid rationales, we apply a *correctness-based filter*:

$$\mathcal{D}_{distill}^{(role)} = \left\{ (x, \hat{r}, \hat{y}) \,\middle|\, (\hat{r}, \hat{y}) = \text{LLM}_T(x, p_{role}),\ \hat{y} = y_{gt} \right\} \quad (3)$$

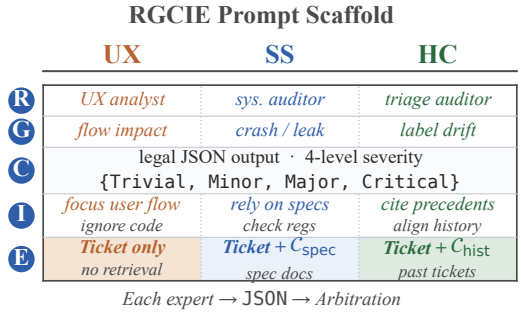


Figure 4: RGCIE prompt scaffold. Merged *C* row: shared output contract; highlighted *E* row: per-expert retrieval.

where x is the ticket, p_{role} is the role-specific prompt, $\hat{r}$ is the reasoning trace, and y_{gt} is the ground-truth severity. In practice, the filter admits roughly 80% of teacher-generated traces across the three roles, retaining a sufficient number of high-fidelity samples for adapter training while discarding the 20% in which teacher reasoning leads to an incorrect verdict. The purpose of this filter is specifically to align each adapter with a *valid role-specific reasoning pattern*: teacher mistakes, if distilled into the adapter, would bake flawed reasoning into the expert weights and be structurally hard to correct downstream. We therefore deliberately keep failure-mode learning *out* of the adapter stage and defer it to the decision module (Section 3.2), whose training set explicitly includes tickets where individual experts err—so that the module learns *when* to trust or discount each expert rather than being trapped by confidently wrong expert outputs.

Parameter-Efficient Role Adaptation. We fine-tune three **LoRA adapters** [27] on a shared `DeepSeek-R1-7B` student backbone using $\mathcal{D}_{distill}$. This multi-adapter design realizes DR1 and DR3 jointly: the three disjoint adapter subspaces structurally isolate UX, SS, and HC reasoning, preventing the cross-perspective entanglement identified in Challenge 3; sharing the base parameters θ_{base} allows all three experts to run sequentially on a single backbone at

inference time, with LoRA adapters hot-swapped between calls—collectively staying within the 1.8 s budget. The output of this stage is a shared backbone θ_{base} and three role-specific adapters $\{\Delta\theta_{UX}, \Delta\theta_{SS}, \Delta\theta_{HC}\}$.

3.2 Offline Decision Module Training

The second offline phase trains a **Ticket-Anchored Decision Module** to realize DR2. The module conditions expert fusion on the semantics of the incoming ticket rather than on fixed weights [28].

3.2.1 Feature Representation. The module is trained on the full labeled set $\mathcal{D}_{train}$ without correctness filtering; each instance x is paired with the pre-generated outputs from the student expert ensemble $\{(\hat{r}_i, \hat{y}_i)\}_{i=1}^{3}$, including tickets on which individual experts produce wrong verdicts.

- **Ticket Anchor:** The raw ticket is projected into a latent **ticket anchor** $h_x = \text{Enc}(x) \in \mathbb{R}^d$, capturing its global semantic context.
- **Expert Signal:** For each expert $i \in \{\text{UX, SS, HC}\}$, the reasoning trace and verdict are combined into a composite representation:

$$h_i = \mathbf{W}_E \left[\text{Enc}(\hat{r}_i) \,\|\, \text{Embed}(\hat{y}_i)\right] \tag{4}$$

where $\|$ denotes concatenation, and $\mathbf{W}_E$ is a trainable projection that maps heterogeneous expert signals into a unified embedding space.

3.2.2 Ticket-Anchored Attention. Fusion is computed via attention with the ticket anchor as *Query* (Q) and expert representations as *Keys* (K) and *Values* (V). The attention weight α_i encodes the learned reliability of expert i for the given ticket:

$$\alpha_i = \frac{\exp\left(\frac{(\mathbf{W}_Q h_x)^\top (\mathbf{W}_K h_i)}{\sqrt{d_k}}\right)}{\sum_{j=1}^{3} \exp\left(\frac{(\mathbf{W}_Q h_x)^\top (\mathbf{W}_K h_j)}{\sqrt{d_k}}\right)} \tag{5}$$

where d_k is the projection dimension, and $\mathbf{W}_Q, \mathbf{W}_K$ are learnable. This lets the module prioritize experts whose reasoning aligns with the ticket's semantic signature—for example, favoring SS on technical kernel panics and UX on subjective UI-latency issues.

3.2.3 Training and Calibration. The fused representation $h_{\text{fuse}} = \sum_{i=1}^{3} \alpha_i \mathbf{W}_V h_i$ feeds a classifier $\hat{Y} = \text{Softmax}(\text{MLP}(h_{\text{fuse}}))$, optimized against ground-truth labels via cross-entropy loss $\mathcal{L}_{CE}$. During training, the **student backbone and LoRA adapters remain frozen**; only the decision module is updated. This division of labor is deliberate. The adapters are aligned to valid reasoning patterns by $\mathcal{D}_{distill}$—which inherently biases the experts toward overconfidence, including on inputs where the underlying reasoning is subtly unreliable. The decision module, in contrast, is trained on the *full* labeled distribution with no correctness filter, so it directly observes cases where individual experts fail and must learn the conditions under which each expert's verdict should be discounted. In effect, the two stages split the learning objective cleanly: the LoRA adapters learn *how to reason*, while the arbitration module learns *when to trust each reasoner*—a separation that is especially important under the release-driven drift of Section 2.2, where the conditions triggering expert failure change between releases. The output is a trained decision module ready for deployment.

3.3 Online Inference and Triage

Given an incoming ticket x, the three experts run sequentially on the shared `DeepSeek-R1-7B` backbone θ_{base} with LoRA adapters $\Delta\theta_i$ hot-swapped between calls, after which the decision module produces the final severity label Y^*.

3.3.1 Retrieval-Augmented Expert Execution. The UX expert reasons directly over x without retrieval. The SS and HC experts operate in retrieval-augmented mode [29]: a semantic query is issued over the offline corpora,

$$\mathcal{K}_{context} = \text{Retrieve}(x, C_{spec}) \cup \text{Top-}k(x, C_{hist}), \tag{6}$$

and the retrieved technical specifications and historical precedents are injected into the respective expert prompts. End-to-end execution, including adapter switching and retrieval, stays within the 1.8 s latency budget.

3.3.2 Decision Fusion. The expert outputs $\{(\hat{r}_i, \hat{y}_i)\}_{i=1}^{3}$ are passed to the Ticket-Anchored Decision Module (Section 3.2.2), which encodes the ticket and expert signals, computes attention weights α_i, and emits the fused label Y^* in a single forward pass. Y^* is then forwarded to the production triage workflow for prioritization and engineer assignment.

4 Evaluation

In this section, we conduct a comprehensive evaluation of *MERA* to answer the following research questions (RQs):

- **RQ1:** How does *MERA* perform overall on industrial ticket severity assessment compared with representative baselines?
- **RQ2:** Does each key component of *MERA* contribute significantly to its performance?
- **RQ3:** How well does *MERA* generalize to unseen and evolving ticket patterns in practical deployment scenarios?

4.1 Experiment Setup

4.1.1 Dataset. We conduct experiments on an **expert-curated golden set** of 6,155 tickets, sampled via semantic clustering from 370,000 production records to cover diverse failure modes in Huawei's beta environment. Senior triage engineers performed **multi-round manual verification** on all labels, prioritizing **label correctness** over raw volume. Severity follows industrial practice and is categorized into four ordered classes: Trivial, Minor, Major, and Critical. We split the set into 5,505 training and 650 test tickets. From the 5,505 training tickets, our multi-perspective decomposition generates **16,515 reasoning traces** (three role-specific traces per ticket); after the correctness-based filter of Section 3.1, roughly 80% (~13,200) form the distillation set $\mathcal{D}_{distill}$—a scale well-suited for **domain alignment**, where logic-rich samples matter more than raw volume [30].

The dataset exhibits a **natural class imbalance** (Table 2), particularly the scarcity of Critical defects. We therefore supplement accuracy with confusion matrices and the **Ordinal Mean Absolute Error (MAE_{ord})** for calibrated assessment across all severity levels.

4.1.2 Implementation Details. All experiments run on a Huawei server with four Ascend 910B NPUs (PyTorch 2.5.1, CANN 8.1). We use `DeepSeek-R1-32B` [24] as the offline teacher and `DeepSeek`

Table 2: Distribution of severity levels in the Huawei industrial ticket dataset.

Severity Level	Training Set	Test Set
Trivial	199	40
Minor	2,865	345
Major	2,164	241
Critical	277	24

-R1-7B as the deployable student, with LoRA [27] fine-tuning implemented via PEFT and DeepSpeed ZeRO-3. For baselines and ablations, we also train a label-supervised LoRA SFT variant on the 7B backbone; applying the same SFT to the 32B teacher exceeds device memory even with ZeRO-3, so the 32B model is kept frozen throughout and used only for trace generation and as a teacher-scale inference reference.

4.1.3 Metrics. We report four complementary metrics. **Accuracy** summarizes overall correctness under natural class frequencies. Given the dataset's substantial class imbalance (Critical and Trivial are far rarer than Minor/Major), we additionally report **macro-averaged Precision, Recall, and F1-score** to equally weight rare but business-critical severities. Finally, since severity is ordinal (Trivial < Minor < Major < Critical), we report the **ordinal Mean Absolute Error** (MAE_{ord}) over integer-coded labels $\{0, 1, 2, 3\}$, which penalizes large mis-rankings (*e.g.*, predicting Minor for a Critical ticket).

4.1.4 Baselines. Prior to *MERA*, ticket severity at Huawei was assessed entirely by manual triage; no automated grading system was in production. To evaluate *MERA* comprehensively, we select baselines from two distinct paradigms: (1) **Discriminative Classifiers**, representing the established industrial standard for cost-effective automated triage; and (2) **Generative LLM approaches**, representing the state-of-the-art in semantic reasoning. To ensure a rigorous comparison, we performed grid search over each baseline's principal hyperparameters on the validation set— including learning rate, batch size, and method-specific knobs (e.g., LoRA rank for SFT variants, prototype count for CPM-Hier, retrieval depth for RAG-augmented baselines)—and report the best validation configuration for each method.

- **AlertRank [15]:** An unsupervised alert triage framework that learns priority rankings from historical handling patterns. We adapt its scoring to ticket texts enriched with statistical features (device model, build version, keyword frequencies).
- **PLM+CE [31]:** A BERT-style pre-trained language model fine-tuned for severity classification with standard cross-entropy loss. Represents the flat-classification baseline among PLM-based approaches.
- **PLM+CORN [32]:** A PLM fine-tuned with Conditional Ordinal Regression (CORN), which explicitly models the ordinal structure Trivial < Minor < Major < Critical rather than treating severities as independent classes.
- **CPM-Hier [17]:** A hierarchical class-prototype model that learns severity-level prototypes and classifies tickets by prototype similarity, representing the state-of-the-art text-classifier approach for bug triage.
- **DeepSeek-Inf (32B/7B + RAG):** Evaluates retrieval-augmented LLMs without parameter updates, as a monolithic alternative to the multi-expert architecture. `DeepSeek-R1-32B/7B` receives the ticket enriched with retrieved context and produces a severity label via a single unified prompt, with no reasoning decomposition across roles.
- **DeepSeek-SFT (7B + RAG):** This baseline emerged from our early development iterations as the most direct LLM-based approach. It fine-tunes a monolithic `DeepSeek-R1-7B` via LoRA on labeled tickets with the same retrieval corpus as *MERA*, mapping retrieved context directly to severity labels without reasoning decomposition or dynamic arbitration.

4.2 RQ1: Overall Effectiveness

Table 3 reports the overall comparison. *MERA* achieves an $F1_{macro}$ of 0.9060, outperforming all baselines. AlertRank performs poorly in this telemetry-void setting, and text classifiers such as CPM-Hier, though competitive in-distribution, degrade under drift (discussed in Section 4.4). Among generative baselines, label-supervised fine-tuning (DeepSeek-SFT, 7B) reaches only 0.5907—a 31.5% gap to *MERA*—suggesting that mapping symptom descriptions to severity levels is difficult without intermediate reasoning. Our 7B student also exceeds the 32B inference-only model (DeepSeek-Inf, 0.8080), indicating that the distilled reasoning traces carry information a monolithic LLM does not recover even at larger scale.

Figure 5 shows that most errors fall on adjacent severity levels, with the Minor↔Major boundary being the dominant source. Only 6 of 650 predictions deviate by more than one ordinal step, consistent with an MAE_{ord} of 0.0969. Recall on Critical reaches 87.5%, which matters in deployment because under-grading a release-blocking defect is the most costly failure mode.

Despite per-method hyperparameter tuning, baselines fail along two recurring axes—pattern-matching brittleness under drift and monolithic perspective entanglement—confirming that neither scaling nor fine-tuning substitutes for structural decomposition.

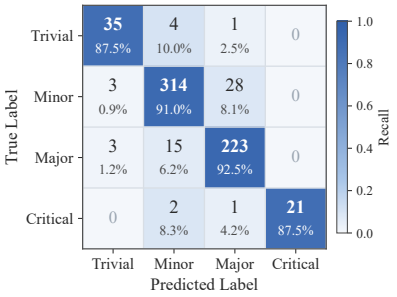


Figure 5: Normalized confusion matrix of *MERA* on the test set. Cell values show raw counts and row-normalized percentages.

4.3 RQ2: Ablation Study

To assess the contributions of individual components in *MERA*, we conduct ablation studies focusing on the training strategy, inference architecture, and decision mechanism. Table 4 presents the comprehensive results.

4.3.1 Impact of Reasoning-Trace Distillation (DR3). To isolate the effect of training strategy, we keep the full multi-expert + ticket-anchored fusion architecture intact and vary only how the three 7B experts are trained:

Table 3: Overall performance comparison on industrial ticket severity assessment. Bold indicates the best performance.

Method	Acc.	P_{macro}	R_{macro}	$F1_{macro}$	MAE_{ord}
Discriminative Classifiers (PLM/Statistical)					
AlertRank [15]	0.7785	0.8836	0.5527	0.6066	0.2231
PLM+CE [31]	0.8462	0.7909	0.8051	0.7975	0.1769
PLM+CORN [32]	0.8677	0.8648	0.8472	0.8549	0.1415
CPM-Hier [17]	0.8800	0.8868	0.8106	0.8425	0.1338
Generative LLM Baselines (DeepSeek-R1)					
DeepSeek-Inf (7B+RAG)	0.7338	0.9172	0.5366	0.6019	0.5754
DeepSeek-Inf (32B+RAG)	0.8092	0.8432	0.7901	0.8080	0.2031
DeepSeek-SFT (7B+RAG)	0.7138	0.7977	0.5207	0.5907	0.6200
MERA (Ours)	**0.9123**	**0.9181**	**0.8964**	**0.9060**	**0.0969**

Table 4: Comprehensive ablation study of *MERA*. We evaluate contributions across training, inference, and decision dimensions.

Variant	Acc.	P_{macro}	R_{macro}	$F1_{macro}$	MAE_{ord}
Full System					
***MERA* (7B)**	**0.9123**	**0.9181**	**0.8964**	**0.9060**	**0.0969**
1. Training Strategy Ablation					
7B w/o SFT	0.8754	0.8768	0.8465	0.8584	0.1292
7B w/ Label SFT	0.8846	0.8792	0.8751	0.8765	0.1231
32B w/o SFT	0.9092	0.9311	0.8894	0.9087	0.0954
2. Inference Architecture Ablation					
Mono-7B (Single)	0.7727	0.8572	0.6190	0.6684	0.2464
Mono-32B (Single)	0.8092	0.8432	0.7901	0.8080	0.2031
3. Decision Mechanism Ablation					
w/ HardVote (Majority)	0.5353	0.5083	0.4771	0.4252	0.5836
w/ CDST (Stat. Weight)	**0.8204**	**0.8064**	**0.7910**	**0.7978**	**0.2011**

- **7B w/o SFT:** Three experts share a frozen DeepSeek-R1-7B backbone with no task adaptation; retrieval and decision fusion remain active.
- **7B w/ Label SFT:** Experts are fine-tuned with LoRA on ticket-to-label pairs only (standard supervised classification), no reasoning traces.
- **32B w/o SFT:** Three experts share a frozen DeepSeek-R1-32B backbone (teacher-scale, no LoRA); serves as an upper-bound reference.
- ***MERA* (7B):** Three experts fine-tuned with LoRA on the filtered teacher reasoning traces (our full design).

As listed in Table 4, the distilled 7B student ($F1_{macro}$=0.9060) matches the teacher-scale 32B frozen ensemble within 0.3% on $F1_{macro}$ and Accuracy (0.9060 vs 0.9087). Compared with label-only SFT on the same backbone ($F1_{macro}$=0.8765), trace supervision brings a 2.95-point improvement, confirming that the student benefits from learning *how* to reason rather than only the end label. Crucially, this reasoning quality comes at deployable cost. As detailed in Table 5, the 32B ensemble requires 2.07 s per ticket and violates the 1.8 s budget, whereas the trace-distilled 7B ensemble reduces total wall time to 1.48 s (−28.5%) and median inference latency by 38.1%. Trace distillation therefore retains near-teacher reasoning quality while meeting the online latency constraint (Section 2.1).

4.3.2 Impact of Multi-Expert Decomposition (DR1). The inference module decomposes severity grading into three complementary perspectives. We evaluate this design from two angles: whether decomposition itself helps, and whether all three perspectives are necessary.

Table 5: Deployment efficiency comparison between a teacher-scale 32B deployment and a trace-distilled 7B deployment (offline batch inference).

Metric	32B	7B (Trace)	Δ (7B vs 32B)
Median E2E latency per record (ms) ↓	737.38	456.50	−38.1%
Mean E2E latency per record (ms) ↓	717.71	452.50	−37.0%
Mean LLM latency per record (ms) ↓	643.53	377.74	−41.3%
LLM throughput (tokens/s) ↑	1926.28	2476.28	+28.6%
LLM throughput (records/s) ↑	2.25	2.65	+17.8%
Total wall time per ticket (s) ↓	2.07	1.48	−28.5%

Notes: Metrics are collected in an offline batch setting (Ascend 910B, TP=4, batch size=400). Each record corresponds to one expert-role inference.

Table 6: Expert-level ablation: performance of two-expert combinations with fusion module retrained.

Expert Combination	Acc	P_{macro}	R_{macro}	$F1_{macro}$	MAE_{ord}
UX+SS+HC (full MERA)	0.9123	0.9181	0.8964	0.9060	0.0969
UX+SS	0.6615	0.6492	0.5481	0.5451	0.4431
UX+HC	0.6354	0.6683	0.5835	0.5596	0.4585
SS+HC	0.7031	0.6776	0.6544	0.6575	0.3954

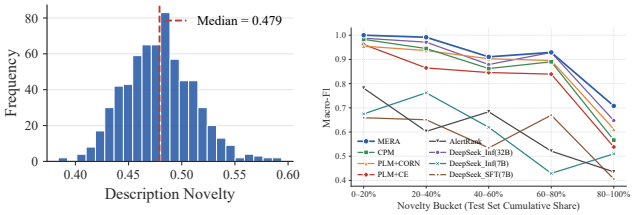
We first compare *MERA* with single-model baselines that retain all three perspectives in one reasoning chain. To isolate the architectural contribution from training supervision, Mono-7B is fine-tuned on the *union* of all three teacher trace sets via a single LoRA adapter and performs single-pass inference—matching *MERA*'s training data while collapsing its multi-expert architecture; Mono-32B remains frozen (Section 4.1.2) as a teacher-scale reference. Both fall short of the full three-expert model ($F1_{macro}$ 0.6684 and 0.8080 vs. 0.9060, Table 4), confirming that perspective entanglement is a structural barrier that neither scaling nor shared trace supervision can resolve.

We then ask whether two perspectives suffice. For each pair, we retrain the fusion module on the retained experts. As Table 6 shows, every two-expert variant degrades substantially (best $F1_{macro}$ 0.6575 for SS+HC vs. 0.9060), and even falls below the monolithic Mono-7B baseline—an absent perspective cannot be recovered by re-routing attention across the remaining two. Each expert therefore contributes information the others do not, justifying the three-expert design.

4.3.3 Impact of Ticket-Anchored Arbitration (DR2). The Ticket-Anchored Decision Module employs an attention mechanism to dynamically weigh expert perspectives, resolving conflicts based on ticket-specific semantics. To evaluate the necessity of this semantic arbitration, experiments are conducted against two representative strategies: a rigid majority rule (denoted as w/ HardVote) and a statistical ensemble method based on covariate-dependent stacking (denoted as w/ CDST) [33]. As listed in Table 4, the rigid w/ HardVote baseline results in a substantial performance drop ($F1_{macro}$ 0.4252), confirming that fixed priority rules are unreliable when experts diverge. While the statistical w/ CDST approach significantly improves over voting ($F1_{macro}$ 0.7978), it remains inferior to *MERA* ($F1_{macro}$ 0.9060). The 10.8-point gap between *MERA* and CDST ($F1_{macro}$ 0.9060 vs 0.7978) confirms that fusion quality depends on conditioning the weighting on ticket semantics, not merely on the statistics of expert outputs.

4.4 RQ3: Generalization to Unseen Ticket Descriptions

Real-world deployment continuously surfaces tickets with user descriptions that diverge from training patterns (*e.g.*, due to new feature launches). To assess operational stability, we evaluate performance under out-of-distribution (OOD) descriptions, focusing on semantic novelty in the ticket text.



(a) Novelty Distribution (b) Robustness (Macro-F1)

Figure 6: Novelty distribution and model resilience. (a) Long-tail distribution of semantic novelty in test tickets. (b) Performance across novelty buckets.

4.4.1 Characterizing Unseen Descriptions. We quantify the "novelty" of a test ticket based on its semantic distance to the training distribution. For each test ticket x, we compute the maximum cosine similarity to the training set: $s(x) = \max_{x' \in \mathcal{D}_{train}} \cos(e(x), e(x'))$. We define $novelty(x) = 1 - s(x)$ [34], where higher values indicate descriptions semantically distant from historical precedents. Figure 6a illustrates the novelty distribution, confirming that a non-trivial portion of incoming tickets constitutes OOD examples.

4.4.2 Performance under Novelty Shift. We partition the test set into K buckets by description novelty (from In-Distribution to OOD) and report per-bucket performance. Figure 6b summarizes the results. As novelty increases, discriminative baselines (*e.g.*, PLM+CE) suffer a sharp performance collapse, indicating reliance on surface-level keyword memorization. In contrast, *MERA* exhibits **graceful degradation**, maintaining the highest $F1_{macro}$ even in the most novel buckets. This "flat" degradation curve demonstrates that *MERA* retains grading reliability even when facing unseen user phrasing.

4.4.3 Analysis of Robustness Mechanisms. *MERA*'s resilience stems from three sources acting together: distilled LLM reasoning generalizes beyond surface keywords, retrieval anchors judgments to stable specifications, and the ticket-anchored fusion recalibrates expert weights as their divergence grows under novelty—together preserving decision boundaries despite semantic drift.

5 Discussion

5.1 Deployment

MERA has been integrated into Huawei's beta-ticket pipeline for five months, processing over 370,000 tickets with daily volume ranging from 5,000 to 10,000 during active release windows. As illustrated in Figure 7, the system serves as a **fully automated first-pass grader**: every incoming ticket receives a severity label and a structured three-perspective rationale within seconds, without human intervention. Triage engineers validate production quality through random spot-checks on ~5% of daily volume. Over this

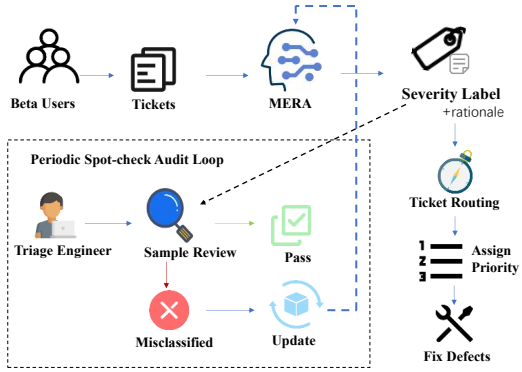


Figure 7: Deployment of *MERA* in the ticket handling pipeline.

period, *MERA* has sustained **85% production accuracy**—a modest drop from the 91.2% offline result that is concentrated on adjacent Minor↔Major boundaries rather than on the most costly failure mode: across the ~4% of production tickets graded Critical, recall remains around **90%**, on par with the offline 87.5%. Release-blocking defects are therefore captured with comparable reliability in deployment. *MERA* has shifted the team from line-by-line manual triage to spot-check auditing and downstream analysis, reducing overall triage labor by approximately **60%**—equivalent to roughly **18 engineer-days saved per major HarmonyOS release**. Across ~18,400 audited tickets, engineers overrode *MERA*'s verdict on 15.0% of cases, with residual errors skewing toward under-grading and concentrated on adjacent severity levels.

Release-Aware Maintenance. The inference server maintains a single frozen 7B backbone with dynamically switched Multi-LoRA adapters—the same sequential-adapter design of Section 3.3—which reduces the NPU memory footprint by **62%** relative to deploying three separate 7B expert models. This architecture enables **zero-downtime maintenance** aligned with Huawei's release cadence of every two to three weeks: knowledge bases are rebuilt each cycle (~2 hours), and expert adapters are fine-tuned monthly from fresh teacher traces (~4 NPU-hours, off-peak). Figure 8 traces production accuracy across seven release cycles. Each new OS build may introduce previously unseen vocabulary and failure modes, causing a transient accuracy dip; the subsequent KB rebuild re-indexes updated documentation and typically restores accuracy to the 84–86% target band within days. For larger feature drops that reshape failure-mode distributions more substantially, the monthly adapter retrain recalibrates the expert modules on fresh teacher traces for a deeper correction. Several releases (R2, R4, R7) incur no observable degradation, indicating that the frozen backbone generalizes well to incremental updates within the existing failure-mode vocabulary. Over 7 KB rebuilds and 5 adapter retrains, production accuracy has remained within 2 percentage points of its initial level.

Failure Handling and Generalizability. At 85% production accuracy, roughly one in six sampled tickets requires correction. When such cases arise, the three-expert rationale enables immediate diagnosis: engineers trace the failure to a specific perspective and apply a *hotfix patch* to the relevant KB entry. Patches accumulate between scheduled maintenance windows; at each bi-weekly KB refresh, all outstanding patches are consolidated into a full-index rebuild, preventing patch-over-patch fragmentation. This two-tier refresh—daily hotfix patches coupled with bi-weekly consolidated

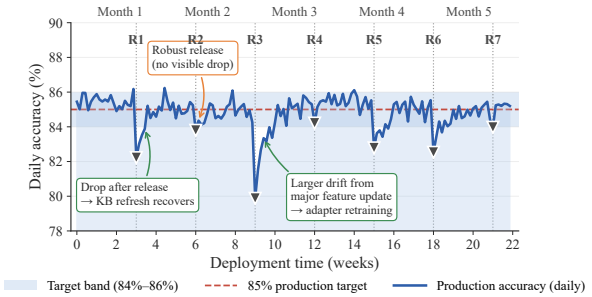


Figure 8: Production accuracy of *MERA* over five months. Vertical dashed lines mark OS release cycles (R1–R7); the shaded band denotes the 84–86% target range. Post-release dips are recovered by KB rebuilds or monthly adapter retrains; several releases cause no observable degradation.

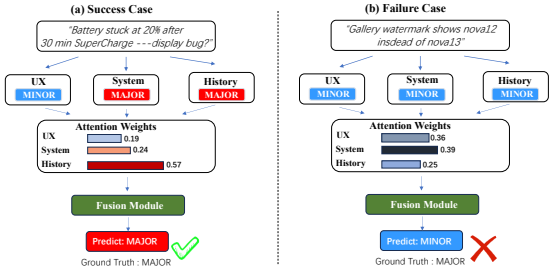


Figure 9: Workflow of a success case and a failure case.("insdead" in the failure case ticket is a user typo, reproduced verbatim to illustrate real-world input noise.)

rebuilds—has, over five months, produced 30+ patches absorbed into 7 scheduled rebuilds, with **no production incident attributed to *MERA*'s misclassification**. We currently rely on these indirect signals—override rate, incident absence, and sustained 5-month adoption—as evidence of engineer trust; a structured rationale-trust survey is planned for the next release cycle.

5.2 Case Study

We present two representative cases to illustrate both the strengths and limitations of *MERA*.

Success Case: Perspective decoupling recovers severity from conflicting signals. Figure 9 illustrates a ticket in which a user reports that their battery remains at 20% after 30 minutes of SuperCharge, suspecting "*just a display bug*." A monolithic LLM baseline anchors on the non-urgent framing and grades this as Minor. Under *MERA*, the three experts diverge productively: the UX expert, following the user's calm tone, reads it as a non-blocking UI synchronization issue (Minor); the SS expert cross-references the PowerManager specification and flags the stagnation as a charging-subsystem functional failure (Major); the HC expert retrieves analogous tickets consistently graded Major or above. Conditioned on the ticket's charging-related semantics, the ticket-anchored fusion module directs attention toward the technically grounded evidence, yielding attention weights of 0.57 (SS), 0.24 (HC), and 0.19 (UX), and arbitrates the conflict in favor of Major. This case demonstrates the core value of perspective decoupling: when user sentiment and technical risk point in opposite directions, structural separation prevents the former from masking the latter.

Failure Case: Decoupling cannot compensate for uniformly misleading surface cues. A user reports: "*After taking a photo, the watermark in the gallery shows nova12 instead of nova13.*" The ground truth is Major—a device model identifier error implies a system-level configuration fault that could affect OTA delivery, warranty validation, and downstream services. However, *MERA* predicts Minor. The UX expert finds no emotional distress and treats it as a cosmetic text error. The SS expert, whose RAG retrieval is steered by the ticket's surface semantics ("watermark," "gallery"), retrieves rendering-related component knowledge and associates the issue with the gallery application's display logic rather than with low-level system properties. The HC expert retrieves past watermark-related tickets, which were predominantly labeled Minor. Because every expert's evidence channel—sentiment, component retrieval, and historical analogy—is anchored to the same misleading surface cue, all three verdicts independently land on Minor. Attention weights of 0.39 (SS), 0.36 (UX), and 0.25 (HC) redistribute the vote but not the consensus: with no expert dissenting, the fusion module has no Major signal to up-weight, and the final output is Minor.

This case exposes a structural limit of *MERA*: multi-expert decomposition protects against *inter-channel* interference (one perspective being overridden by another), but it cannot recover from *uniform* bias where every channel is misdirected by the same surface cue. Fusion has nothing to up-weight when all three experts are anchored to the same misleading feature. This is precisely why *MERA* is deployed as an automated first-pass grader rather than a fully autonomous decision maker: the spot-check audit and two-tier KB refresh described in Section 5.1 are designed to absorb exactly these cases. Once an engineer flags a misgrading like the "nova12/nova13" ticket during auditing, a hotfix patch is added to C_{spec} linking the watermark component to device-identifier consistency checks; subsequent tickets with similar surface cues are then correctly retrieved and graded. Uniform-bias failures are therefore not an unaddressed blind spot but a predictable class of cases that the human-in-the-loop maintenance cycle is explicitly built to catch and correct.

5.3 Lessons Learned

5.3.1 Sentiment misleads; prompting cannot fix it—structure can. An early single-model prototype consistently over-graded tickets where users expressed strong frustration about non-critical components—urgent language about a minor UI glitch would be escalated to Major because emotional intensity is more lexically salient than technical specifics. Prompt-level interventions (e.g., "disregard emotional tone, focus on technical impact") proved unreliable; compliance was inconsistent across tickets and prompt iterations. The failure disappeared only after we moved to architecture-level isolation, assigning each reasoning dimension to an independent expert. The lesson generalizes: in any triage task where subjective tone coexists with objective evidence, a single reasoning chain lets the louder signal dominate the more important one, and prompt-level interventions alone cannot enforce the separation that structural decoupling provides.

5.3.2 Distilled reasoning has longer shelf life than the context it operates on. We initially planned to retrain expert adapters after

every OS release, following classifier-based triage pipelines where model drift tracks vocabulary drift one-to-one. Production monitoring revealed a different dynamic: refreshing the retrieval corpora alone, without any LoRA update, preserved accuracy across consecutive release cycles, and adapter retraining was relaxed to monthly cadence with negligible accuracy variation. The deeper lesson is that once adapters have internalized stable role-specific inference patterns, the distilled reasoning ages slowly; what ages quickly is the factual context it operates on (new APIs, device models, failure modes), which lives in the retrieval corpora. Inverting the maintenance priority—KB-first, model-second—reduces each bi-weekly release from several NPU-hours of retraining to minutes of re-indexing, and may generalize to other distilled LLM systems whose reasoning procedure and domain knowledge evolve on different timescales.

5.4 Generalizability and Scalability

Generalizability. *MERA*'s architecture is intended as a generic triage scaffold: the organization-specific elements (severity rubric, retrieval corpora, and expert role set) are pluggable, while the architectural invariants—multi-expert decomposition, teacher–student distillation, and ticket-anchored arbitration—carry no dependence on Huawei's labeling conventions. Adapting *MERA* to other domains (automotive OS beta, IoT firmware, enterprise software pre-release) requires only replacing the corpora and re-running distillation on local teacher traces. More broadly, any triage setting whose evidence factors along complementary axes—subjective, objective, and precedent-based—fits the same scaffold, with expert roles tailored to the domain.

Scalability. Under current production load, the distilled 7B ensemble sustains 2.65 records/s per NPU instance (Table 5) at 1.48 s per-ticket latency, with throughput scaling linearly via additional instances—sufficient headroom beyond the current 10,000/day ceiling. Extending the expert set adds only one LoRA adapter per new role on the shared backbone, with linear cost in memory and latency, so the role inventory can evolve with organizational priorities without re-architecting the system.

5.5 Threats to Validity

- *Scarcity of Critical samples.* The natural rarity of Critical defects constrains the statistical power of our test-set evaluation (24 instances). Per-class metrics for this category remain strong, but we acknowledge the limited statistical confidence. We draw supplementary assurance from production deployment: all Critical tickets undergo mandatory human review, and no production incident has been attributed to *MERA*'s misclassification over the five-month observation period.
- *Single-organization deployment.* Our evaluation is conducted exclusively within Huawei's mobile OS beta pipeline, and the severity definitions, annotation conventions, and ticket language style reflect this organizational context. *MERA*'s architecture—expert decomposition, distillation pipeline, and ticket-anchored fusion—is designed to be decoupled from any particular knowledge base or labeling convention, requiring only KB replacement and re-distillation to adapt; however, we have not yet validated the framework across other organizations, product domains, or ticket languages. Generalizability to different triage contexts remains an open question for future work.

6 Related Work

6.1 Ticket Severity Assessment

Prior work on severity assessment has progressed from text mining and classical classifiers [35, 36] to neural encoders—CNNs, LSTMs, and pre-trained Transformers like BERT—that learn richer semantic representations [31, 37], with some approaches further exploiting relational structure via graph embeddings [38]. These methods function as static pattern matchers tied to stable vocabularies or auxiliary telemetry, and become brittle in the telemetry-void beta setting where ticket semantics drift with each release. *MERA* departs from surface-level classification by arbitrating decoupled expert reasoning through a ticket-anchored fusion module.

6.2 LLMs for Software Engineering

Recent LLM applications in SE span autonomous repair (SWE-agent, AutoCodeRover, Agentless, OpenHands) [39–42], code-aware retrieval (SWE-Search, CodeRAG) [43, 44], and QA automation such as fault localization and test generation [45, 46]. These efforts predominantly target code artifacts through online agent loops. *MERA* instead targets *human-written ticket triage*, a different data modality, and achieves production-scale deployability by distilling teacher reasoning into lightweight role-specific experts rather than running agentic search at inference time.

7 Conclusion

This paper presented *MERA*, an LLM-augmented severity grading framework for mobile OS beta testing that decomposes ticket assessment into independent expert perspectives and arbitrates their outputs through ticket-anchored dynamic fusion. By distilling teacher-scale reasoning into lightweight online experts served via a shared multi-LoRA backbone, *MERA* reconciles reasoning quality with production latency constraints. Evaluated on Huawei's production data, *MERA* achieves an $F1_{macro}$ of 0.906 and has been deployed for five months, processing over 370,000 tickets with 85% accuracy and reducing overall triage labor by approximately 60%. These results demonstrate its practicality for automated triage in telemetry-starved beta environments, while highlighting the importance of structural decoupling and release-synchronized knowledge updates in sustaining performance under concept drift.

Data Availability Statement

Production data and code are restricted under NDA. A demonstration implementation with an LLM-synthesized reference dataset that preserves the structural and semantic characteristics of original tickets is archived on Zenodo (https://doi.org/10.5281/zenodo.19673527; GitHub mirror: https://github.com/GEROGEZ/MERA).

References

[1] Domenico Cotroneo, Antonio Ken Iannillo, Roberto Natella, and Roberto Pietrantuono. A comprehensive study on software aging across android versions and vendors. *Empirical Software Engineering*, 25(5):3357–3395, 2020.

[2] Jun Gao, Li Li, Tegawendé F Bissyandé, and Jacques Klein. On the evolution of mobile app complexity. In *2019 24th international conference on engineering of complex computer systems (ICECCS)*, pages 200–209. IEEE, 2019.

[3] Yang Li, Hao Lin, Zhenhua Li, Yunhao Liu, Feng Qian, Liangyi Gong, Xianlong Xin, and Tianyin Xu. A nationwide study on cellular reliability: Measurement, analysis, and enhancements. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, pages 597–609, 2021.

[4] Shivi Garg and Niyati Baliyan. Comparative analysis of android and ios from security viewpoint. *Computer Science Review*, 40:100372, 2021.

[5] Xunzhu Tang, Haoye Tian, Pingfan Kong, Saad Ezzini, Kui Liu, Xin Xia, Jacques Klein, and Tegawendé F Bissyandé. App review driven collaborative bug finding. *Empirical Software Engineering*, 29(5):124, 2024.

[6] Saurabh Malgaonkar, Sherlock A Licorish, and Bastin Tony Roy Savarimuthu. Prioritizing user concerns in app reviews–a study of requests for new features, enhancements and bug fixes. *Information and Software Technology*, 144:106798, 2022.

[7] Adarsh Anand, Navneet Bhatt, Deepti Aggrawal, and Ljubisa Papic. Software reliability modeling with impact of beta testing on release decision. In *Advances in reliability and system engineering*, pages 121–138. Springer, 2016.

[8] Syed Farhan Alam Zaidi, Honguk Woo, and Chan-Gun Lee. Toward an effective bug triage system using transformers to add new developers. *Journal of Sensors*, 2022(1):4347004, 2022.

[9] Luiz Alberto Ferreira Gomes, Ricardo da Silva Torres, and Mario Lúcio Côrtes. Bug report severity level prediction in open source software: A survey and research opportunities. *Information and software technology*, 115:58–78, 2019.

[10] Nicolas Bettenburg, Sascha Just, Adrian Schröter, Cathrin Weiss, Rahul Premraj, and Thomas Zimmermann. What makes a good bug report? In *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*, pages 308–318, 2008.

[11] John Anvik, Lyndon Hiew, and Gail C Murphy. Who should fix this bug? In *Proceedings of the 28th international conference on Software engineering*, pages 361–370, 2006.

[12] Jaehyung Lee, Kisun Han, and Hwanjo Yu. A light bug triage framework for applying large pre-trained language model. In *Proceedings of the 37th IEEE/ACM international conference on automated software engineering*, pages 1–11, 2022.

[13] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[14] Junjie Chen, Shu Zhang, Xiaoting He, Qingwei Lin, Hongyu Zhang, Dan Hao, Yu Kang, Feng Gao, Zhangwei Xu, Yingnong Dang, et al. How incidental are the incidents? characterizing and prioritizing incidents for large-scale online service systems. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*, pages 373–384, 2020.

[15] Nengwen Zhao, Panshi Jin, Lixin Wang, Xiaoqin Yang, Rong Liu, Wenchi Zhang, Kaixin Sui, and Dan Pei. Automatically and adaptively identifying severe alerts for online service systems. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pages 2420–2429. IEEE, 2020.

[16] Jiazhen Gu, Chuan Luo, Si Qin, Bo Qiao, Qingwei Lin, Hongyu Zhang, Ze Li, Yingnong Dang, Shaowei Cai, Wei Wu, et al. Efficient incident identification from multi-dimensional issue reports via meta-heuristic search. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 292–303, 2020.

[17] Biqing Zeng, Yihao Peng, Jichen Yang, Peilin Hong, and Junjie Liang. Cpm-based hierarchical text classification. *Journal of Artificial Intelligence Research*, 82:367–388, 2025.

[18] Arvind Kumar Gangwar and Sandeep Kumar. Concept drift in software defect prediction: A method for detecting and handling the drift. *ACM Transactions on Internet Technology*, 23(2):1–28, 2023.

[19] Aske Plaat, Annie Wong, Suzan Verberne, Joost Broekens, Niki Van Stein, and Thomas Back. Reasoning with large language models, a survey. *arXiv preprint arXiv:2407.11511*, 1(2):3, 2024.

[20] Albert Sadowski, JarosŁ Chudziak, et al. Structured decomposition for llm reasoning: Cross-domain validation and semantic web integration. *arXiv preprint arXiv:2601.01609*, 2026.

[21] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

[22] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023.

[23] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE transactions on big data*, 7(3):535–547, 2019.

[24] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[25] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[26] Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8003–8017, 2023.

[27] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022.

[28] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

[29] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

[30] Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems*, 36:55006–55021, 2023.

[31] Asif Ali, Yuanqing Xia, Qasim Umer, and Mohamed Osman. Bert based severity prediction of bug reports for the maintenance of mobile applications. *Journal of Systems and Software*, 208:111898, 2024.

[32] Xintong Shi, Wenzhi Cao, and Sebastian Raschka. Deep neural networks for rank-consistent ordinal regression based on conditional probabilities. *Pattern Analysis and Applications*, 26(3):941–955, 2023.

[33] Tomoya Wakayama and Shonosuke Sugasawa. Ensemble prediction via covariate-dependent stacking. *Statistics and Computing*, 35(6):212, 2025.

[34] Yiyou Sun, Yifei Ming, Xiaojin Zhu, and Yixuan Li. Out-of-distribution detection with deep nearest neighbors. In *International conference on machine learning*, pages 20827–20840. PMLR, 2022.

[35] Bancha Luaphol, Jantima Polpinij, and Manasawee Kaenampornpan. Text mining approaches for dependent bug report assembly and severity prediction. *Int. Arab J. Inf. Technol.*, 19(6):915–924, 2022.

[36] DNA Dissanayake, RAHM Rupasingha, and BTGS Kumara. Automatic bug priority prediction using lstm and ann approaches during software development. 2024.

[37] Myint Theingi and Naw Lay Wah. Automated prediction of bug reports' severity using attention-based convolutional neural network. In *2025 IEEE Conference on Computer Applications (ICCA)*, pages 1–6. IEEE, 2025.

[38] Ao Zhou, Guangliang Liu, and Jinjin Mei. Bug report severity prediction based on text embedding via graph transformer. *Applied Soft Computing*, page 114491, 2025.

[39] John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652, 2024.

[40] Yuntong Zhang, Haifeng Ruan, Zhiyu Fan, and Abhik Roychoudhury. Autocoderover: Autonomous program improvement. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 1592–1604, 2024.

[41] Chunqiu Steven Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. Agentless: Demystifying llm-based software engineering agents. *arXiv preprint arXiv:2407.01489*, 2024.

[42] Xingyao Wang, Boxuan Li, Yufan Song, Frank F Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, et al. Openhands: An open platform for ai software developers as generalist agents. *arXiv preprint arXiv:2407.16741*, 2024.

[43] Antonis Antoniades, Albert Örwall, Kexun Zhang, Yuxi Xie, Anirudh Goyal, and William Wang. Swe-search: Enhancing software agents with monte carlo tree search and iterative refinement. *arXiv preprint arXiv:2410.20285*, 2024.

[44] Jia Li, Xianjie Shi, Kechi Zhang, Lei Li, Ge Li, Zhengwei Tao, Fang Liu, Chongyang Tao, and Zhi Jin. Coderag: Supportive code retrieval on bigraph for real-world code generation. *arXiv preprint arXiv:2504.10046*, 2025.

[45] Yihao Qin, Shangwen Wang, Yiling Lou, Jinhao Dong, Kaixin Wang, Xiaoling Li, and Xiaoguang Mao. Agentfl: Scaling llm-based fault localization to project-level context. *arXiv preprint arXiv:2403.16362*, 2024.

[46] Wenhan Wang, Chenyuan Yang, Zhijie Wang, Yuheng Huang, Zhaoyang Chu, Da Song, Lingming Zhang, An Ran Chen, and Lei Ma. Testeval: Benchmarking large language models for test case generation. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 3547–3562, 2025.