Open Journal of Transportation Technologies 交通技术, 2026, 15(6), 770-780
Published Online November 2026 in Hans. https://www.hanspub.org/journal/ojtt
https://doi.org/10.12677/ojtt.2026.156066

Hans汉斯

城市轨道交通综合监控运维报文智能解析工具设计与应用

王　宁，杨科鑫，马文广

济南轨道交通集团运营有限公司，客运部，综合维修二中心，山东 济南

收稿日期：2026年9月22日；录用日期：2026年10月29日；发布日期：2026年11月10日

摘　要

城市轨道交通综合监控系统涉及十余个专业子系统，运维报文格式异构、解析依赖人工经验，导致故障定位效率低、准确率难保障，且报文涉及核心保密信息，无法使用联网工具辅助分析。针对上述问题，本文设计并实现了一套轻量化、本地部署的故障报文智能分析工具。该工具构建标准化解析规则库与寄存器地址映射数据集，实现报文专业自动判别与故障精准定位，并封装为单个独立可执行文件(约56 MB)，免安装、双击即用、完全离线运行。实验结果表明，工具单次报文解析平均耗时4.7分钟，较人工解析效率提升约92%，准确率达100%。工具核心知识库基于特定线路的厂商协议、点位表和历史报文构建，具有高度定制化特征，其设计方法可为同类项目提供借鉴。该方案为轨道交通行业提供了一种低成本、安全可控、方法可借鉴的智能解析工具，具有积极的实践推广价值。

关键词

城市轨道交通，综合监控系统，报文智能解析，数据要素，规则引擎

Design and Application of an Intelligent Parsing Tool for Urban Rail Transit Integrated Supervisory Control Operation and Maintenance Messages

Ning Wang, Kexin Yang, Wenguang Ma

Passenger Transport Department, Comprehensive Maintenance Center No. 2, Jinan Rail Transit Group Operation Co., Ltd., Jinan Shandong

Received: September 22, 2026; accepted: October 29, 2026; published: November 10, 2026

文章引用: 王宁, 杨科鑫, 马文广. 城市轨道交通综合监控运维报文智能解析工具设计与应用[J]. 交通技术, 2026, 15(6): 770-780. DOI: 10.12677/ojtt.2026.156066

Abstract

Urban rail transit integrated supervisory control systems involve more than ten professional subsystems, with operation and maintenance messages featuring heterogeneous formats and parsing heavily reliant on manual expertise. This leads to low fault localization efficiency, unsatisfactory accuracy, and the inability to use online tools due to the confidential nature of the messages. To address these issues, this paper designs and implements a lightweight, locally deployed intelligent message parsing tool. The tool builds a standardized parsing rule base and a register address mapping dataset, enabling automatic subsystem identification and precise fault localization. It is packaged as a single standalone executable file (approximately 56 MB) that requires no installation, runs with a double-click, and operates completely offline. Experimental results show that the average parsing time per message is 4.7 minutes, reducing the time by about 92% compared to manual parsing, with an accuracy rate of 100%. The core knowledge base is built based on the manufacturer protocols, point tables, and historical messages of a specific line, and is therefore highly customized; its design methodology can provide reference for similar projects. This solution provides the rail transit industry with a low-cost, safe, controllable, and methodologically referential intelligent parsing tool, offering significant practical value for broader adoption.

Keywords

Urban Rail Transit, Integrated Supervisory Control System, Intelligent Message Parsing, Data Elements, Rule Engine

Copyright © 2026 by author(s) and Hans Publishers Inc.
This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).
http://creativecommons.org/licenses/by/4.0/
Open Access

1. 引言

在城市规模持续扩张与轨道交通网络加速成网的背景下，城市轨道交通系统已逐步成长为诸多大型城市(如北京、上海、济南等)公共交通体系中不可或缺的组成部分[1]-[3]。作为运营业务的核心中枢，综合监控系统(ISCS)承担着对各专业设备全域感知与集中管控的职责，广泛覆盖门禁(ACS)、自动售检票(AFC)、乘客信息(PIS)、视频监控(CCTV)、滑动门(PSD)等十余个专业子系统。该综合监控系统具备双向交互能力：上行可反馈设备状态，下行可接收中心级控制指令；同时，通过多样化的控制器与接口装置，可实现对具体设备的实时监测与远程控制[4] [5]。

在运营维护过程中，综合监控系统持续产生大量运维报文，这些报文是设备状态感知与故障诊断的核心数据载体。然而，当前精准定位故障原因、高效解析报文仍面临四重制约：其一，故障报文涉及设备配置参数、寄存器地址定义等核心保密信息，受企业数据安全管理制度约束，无法使用联网人工智能工具进行辅助分析，且现有运维体系中缺乏本地化、可离线运行的智能辅助手段，导致故障解析仍停留在人工判读阶段。其二，各专业子系统由不同厂商承建，报文格式、字段定义、编码规则各异，数据异构分散且缺乏统一标准，难以形成跨专业的关联分析与综合研判[6]，故障根因往往需要跨专业协同排查方能定位，进一步加剧了解析难度。其三，传统故障定位完全依赖人工经验的逐条排查，解析人员需逐字逐段比对报文内容、手动核对寄存器地址与状态位含义，单次报文解析耗时近 60 分钟，效率低下且处理周期长；而部分专业故障可能直接影响行车组织与运营安全，报文处理速度直接关系到故障响应时效与

企业运营损失，现有模式已难以满足运维实时性要求。其四，不同专业故障特征各异，解析经验长期分散于个人手中，难以沉淀为标准化、可复用的知识资产，导致知识壁垒高、培训成本大，新员工需投入大量时间进行多专业报文解读培训，人才成长周期显著延长，团队整体运维能力提升缓慢。

针对上述问题，本文提出自主研发一套轻量化、本地部署的故障报文智能分析工具。该工具通过对分散的故障数据进行结构化整合与知识沉淀，以数据驱动替代人工经验，实现报文一键解析与故障精准定位，切实推动数据要素在基层运维中的落地赋能。

2. 技术路线

由于综合监控系统中的存在大量专业且报文协议不尽相同，因此需要做大量的前期工作，主要包含构建标准化解析规则库、建立故障特征与寄存器地址映射、开发智能解析算法、轻量化桌面应用封装、性能测试与次案场验证，其中性能测试与现场验证详见第 3 部分，整体技术路线如图 1 所示。

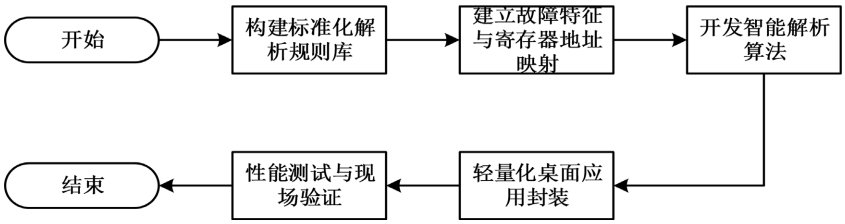


Figure 1. Overall technical route
图 1. 整体技术路线

2.1. 构建标准化解析规则库

系统梳理运维人员在长期故障报文分析中积累的实践经验，逐专业明确报文结构特征、寄存器地址段含义、状态位定义与告警等级分类等核心要素，将人工判读的隐性逻辑显性化为可量化的判别规则，进而构建覆盖多专业、结构清晰、标准统一的报文解析规则库，标准解析规则库如图 2 所示。该规则库

```
// ==================== 地址 0 ====================
addPoint(0, 0, "TDZT1", "异常", "正常", "上行通道数据状态");
addPoint(0, 1, "TDZT2", "异常", "正常", "下行通道数据状态");
addPoint(0, 8, "ZGDGZ", "无故障", "故障", "主攻点系统故障");
addPoint(0, 9, "SKZDYGZ", "无故障", "故障", "上行控制电源故障");
addPoint(0,10, "SQDDYGZ", "无故障", "故障", "上行驱动电源故障");
addPoint(0,11, "XDCGZ", "无故障", "故障", "蓄电池故障");
addPoint(0,12, "XKZDYGZ", "无故障", "故障", "下行控制电源故障");
addPoint(0,13, "XQDDYGZ", "无故障", "故障", "下行驱动电源故障");
addPoint(0,14, "KMGZ", "无故障", "故障", "ASD/EED关门报警故障");
addPoint(0,15, "GMGZ", "无故障", "故障", "ASD/EED关门报警故障");

// ==================== 地址 1 ====================
addPoint(1, 4, "CANZXGZ", "正常", "故障", "CAN总线故障");
addPoint(1, 5, "PEDCGZ", "正常", "故障", "PEDC故障");
addPoint(1, 6, "SYMGBJS", "未激活", "激活", "门关闭/打开状态");
addPoint(1, 7, "SYMWQKQ", "未激活", "激活", "门完全开启状态");
addPoint(1, 1, "KMML", "未触发", "触发", "开门命令");
addPoint(1, 2, "KGMM", "未触发", "触发", "关门命令");
addPoint(1, 3, "MLGZ", "正常", "故障", "门锁故障");
addPoint(1,10, "CZML", "未触发", "允许", "操作命令");
addPoint(1,11, "ZCKM", "未激活", "激活", "中控开门");
addPoint(1,12, "ZCGM", "未激活", "激活", "中控关门");
addPoint(1,14, "FFZCKQ", "未激活", "激活", "防反转开启");
addPoint(1,15, "FFZCGB", "未激活", "激活", "防反转关闭");
addPoint(1, 9, "GZ", "正常", "故障", "综合后备盘故障");

// ==================== 地址 2 ====================
addPoint(2, 0, "HSJCBJ1", "正常", "报警", "滑动门1报警");
```

Figure 2. Standard parsing rule base
图 2. 标准解析规则库

将分散于个人经验的专家知识转化为可复用、可传承的企业显性知识资产，为智能解析算法提供可靠的规则引擎支撑，同时支持新增专业的快速扩展与既有规则的迭代优化。

2.2. 建立故障特征与寄存器地址映射

通过对点位表分析及各专业典型故障案例中报文数据及其对应寄存器状态变化的系统采集与逐条标定，构建了故障特征与寄存器地址的精确映射关系，其中点位表为分析故障特征的核心，点位表如图 3 所示。目前该映射关系已覆盖 ACS、AFC、PIS、CCTV、PSD 等 10 类专业、数百余种故障类型，每类故障均配置多组样本数据，为智能匹配算法提供了可靠完备的数据基础。

名称	链路数量	协议	操作
Z16电力监控系统	2	pscada104(104规约协议)	编辑 删除 关联设备 点位配置
Z16站台门系统	2	modbustcp(标准modbus_tcp驱动协议)	编辑 删除 关联设备 点位配置
Z16火灾自动报警系统	2	modbustcp(标准modbus_tcp驱动协议)	编辑 删除 关联设备 点位配置
Z16环境与设备控制系统	2	modbustcpbas(modbus_tcp_bas)	编辑 删除 关联设备 点位配置
Z16门禁系统	2	modbustcp(标准modbus_tcp驱动协议)	编辑 删除 关联设备 点位配置
Z16自动售检票系统	2	modbustcp(标准modbus_tcp驱动协议)	编辑 删除 关联设备 点位配置
Z16区间感温光纤系统	2	modbustcp(标准modbus_tcp驱动协议)	编辑 删除 关联设备 点位配置
Z16视频监控系统	2	modbustcp(标准modbus_tcp驱动协议)	编辑 删除 关联设备 点位配置
Z16广播系统	2	modbustcp(标准modbus_tcp驱动协议)	编辑 删除 关联设备 点位配置
Z16乘客信息系统	2	modbustcp(标准modbus_tcp驱动协议)	编辑 删除 关联设备 点位配置
Z16能源管理系统	2	pscada104(104规约协议)	编辑 删除 关联设备 点位配置
Z16信号系统	2	sync(point_sync)	编辑 删除 关联设备 点位配置
Z16智能PDU服务器柜	2	SNMP(简单网络管理)	编辑 删除 关联设备 点位配置
Z16智能PDU网络柜	2	SNMP(简单网络管理)	编辑 删除 关联设备 点位配置
Z16安防系统	2	modbustcp(标准modbus_tcp驱动协议)	编辑 删除 关联设备 点位配置

Figure 3. Point address mapping table
图 3. 点位表

2.3. 开发智能解析算法

工具采用“筛选–分组–寻址–取值–查表”五级解析流程。首先根据报文结构特征判别专业类型，并区分指令下发与状态接收场景；随后以 SEND 为边界对 RECV 分组，提取寄存器起始地址(如 SEND 行[00] [09]对应第 15 位寄存器)，计算组内偏移，过滤非必要数据，定位目标数据字；最后提取高低字节转为二进制，与点位表匹配，实现从原始报文到故障定位的全链路映射，如图 4 所示，算法总体流程如图 5 所示。

规则引擎采用“地址–位–语义”三层映射，以 QMap< 寄存器地址，规则列表 > 存储。每条规则含位偏移 bitAddr、点编码 code、0/1 状态描述 desc0/desc1 及扩展说明 extraDesc。解析时先按地址查表，再用(value >> bitAddr) & 1 取位，0 取 desc0、1 取 desc1，输出“点编码(说明)状态”，实现位级故障诊断。

```
[2026-03-25 11:01:02_401] [Z14PA] SEND >>>> tcp头[b1][ed][00][00][00]
[06] 站地址[ff] 功能码[04] 寄存器地址[00][09] 寄存器数量[00][12]  valkind[3]
line[3538]
[2026-03-25 11:01:02_554] [Z14PA] RECV <<<< tcp头[b1][ed][00][00][00]
[27] line[2483]
[2026-03-25 11:01:02_554] [Z14PA] RECV <<<< 站地址[ff] 功能码[04] TCP
头长度[0027] 字节数[0024] line[2698]
[2026-03-25 11:01:02_554] [Z14PA] RECV <<<< total[36] bytes[36]  [15]
[15] [00] [00] [00] [00] [00] [00] [00] [03] [35] [00] [00] [01] [00] [00] [00]
[03] [35] [00] [00] [00] [00] [00] [00] [03] [35] [00] [00] [03] [35] [00] [00]
[03] [35] [00] line[7613]
```

Figure 4. Feature matching diagram
图 4. 特征匹配图

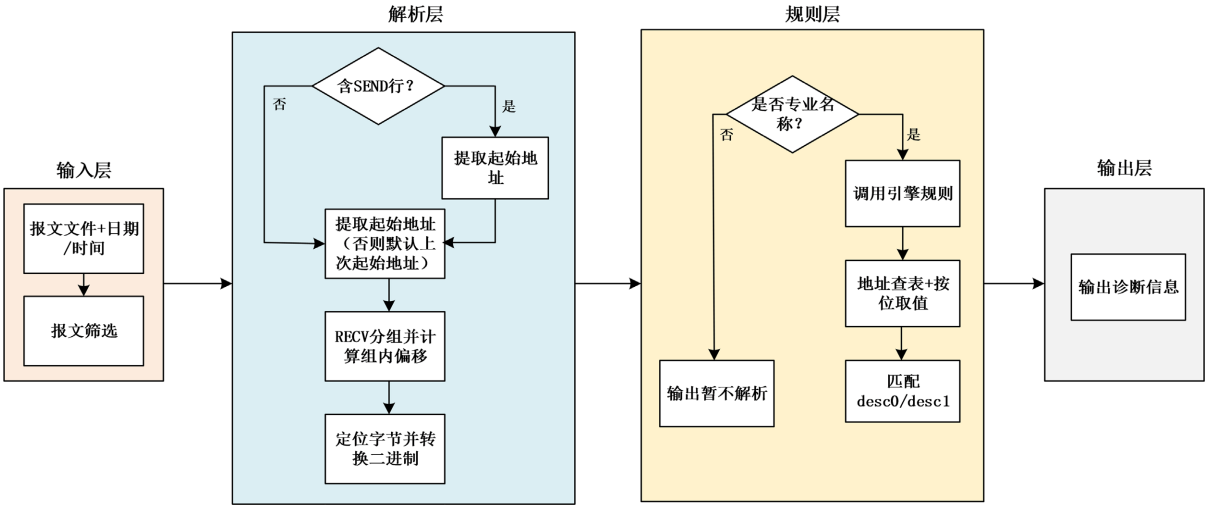


Figure 5. Overall flowchart of the intelligent parsing algorithm
图 5. 算法总体流程图

选用规则引擎而非复杂 AI 模型，主要基于四点：一是报文协议与寄存器定义规则性强，确定性匹配即可满足需求；二是报文涉密，必须完全离线；三是现场终端性能有限，无需训练和 GPU；四是故障定位要求可解释、可追溯，规则引擎优于黑盒模型。

2.4. 轻量化桌面应用封装

工具采用 QT 框架构建轻量化桌面应用，封装为单个可执行文件(约 56 MB)，摒弃了对 Python、数据库等任何第三方运行环境的依赖，实现了免安装、双击即用的极简部署模式，封装后的应用软件大小与界面如图 6 所示；同时工具不包含任何联网功能模块，所有解析处理均在本地终端完成，从架构层面隔绝数据外泄风险。在交互设计上，支持报文文本的拖拽式上传与一键式解析，以图形化操作大幅降低使用门槛，使运维人员无需编程技能即可独立完成报文分析。

3. 实验验证

3.1. 实验场景与设计

为验证故障报文智能分析工具在实际运维场景中的有效性与可靠性，本研究设计了贴近真实业务环境的实验方案。实验以济南轨道交通集团运营一分公司综合监控系统历史运维报文为数据基础，从解析效率、解析准确率与终端兼容性三个维度开展测试，以量化指标评估工具相较于传统人工解析方式的性能提升。



Figure 6. Packaged application software
图 6. 封装后的应用软件

3.1.1. 实验环境

Table 1. Experimental terminal configuration
表 1. 实验终端配置

编号	CPU	内存	操作系统
终端 A	Intel Core i5-7400	8G	Windows 10 专业版
终端 B	Intel Core i5-8400	8G	Windows 10 专业版
终端 C	Intel Core i5-8700	8G	Windows 10 专业版

实验硬件选用 3 台不同配置的办公终端，配置参数如表 1 所示，覆盖基层运维班组的典型终端配置水平。操作系统均为 Windows 10 专业版，实验期间关闭其他应用程序，确保测试结果的独立性与可重复性。

实验工具为自主研发的运维报文故障研判平台，封装为单个 exe 可执行文件(约 56 MB)，不依赖任何第三方运行环境，完全离线运行。

3.1.2. 实验数据集

实验数据来源于综合监控系统历史运维日志，从中随机抽取 80 份故障报文，覆盖 ACS、AFC、PIS、CCTV、PSD 等 10 类专业子系统及数百种故障类型，确保样本在专业分布与故障类型上的代表性。数据集具体构成如表 2 所示。

实验以三位具有五年以上综合监控运维经验的资深工程师的独立解析结果作为准确率判定基准，三人结果一致者直接采纳，不一致者经会商达成共识后作为标准答案。

3.1.3. 评价标准

本研究从以下三个维度评价工具性能：

Table 2. Composition of the experimental dataset
表 2. 实验数据集构成

专业系统	报文数量	故障类型覆盖
PA 广播	8	设备报警状态、控制状态等
AFC 自动售检票	8	闸机故障、票卡读写异常、网络中断等
PIS 乘客信息	6	显示异常、广播故障、时钟同步异常等
CCTV 视频监控	6	视频丢失、存储异常、云台控制故障等
PSD 滑动门	8	开关门异常、防夹触发、通信故障等
BAS 环境与设备监控	6	传感器故障、设备状态异常等
FAS 火灾自动报警	6	探测器报警、系统故障等
SPS 安防	6	液体检测仪设备状态、双鉴探测器状态等
其他专业	26	覆盖其他专业典型故障类型
合计	80	——

(1) 解析耗时：工具对单份报文从导入至结构化结果输出的完整耗时，取 80 份样本的平均值、最大值与最小值，并与传统人工解析耗时进行对比。传统人工基线数据来源于运维班组历史统计记录。

(2) 解析准确率：将工具输出的解析结果(包含故障专业、故障设备、故障类型、寄存器状态变化等关键字段)与专家判定的标准答案进行逐项比对。单份报文全部关键字段与标准答案一致，即判定该报文解析准确。准确率计算公式为：

$$\text{Accuracy} = \frac{\text{TP}}{\text{Total}} \times 100\% \tag{1}$$

其中，Accuracy 为准确率，TP 为正确解析正确报文数量，Total 为总报文数量。

(3) 运行稳定性：工具在 3 台不同配置终端上连续运行，是否出现报错、卡顿或系统异常现象。

3.2. 实验结果与分析

为更客观地验证工具性能，点表用于规则库构建与寄存器地址标定，选用 80 份作为独立测试集，仅用于性能验证，不参与规则库构建。训练集与测试集在时间、设备上无重叠。以下准确率与错误分析均在独立测试集上进行，效率测试在全部 80 份样本上进行。

3.2.1. 解析效率测试

在 3 台测试终端上分别运行工具，对 80 份故障报文逐一完成解析，记录单次耗时并统计各终端上的平均耗时，结果如表 3 所示。

实验结果显示，工具在三台终端上的平均解析耗时分别为 5.2 分钟、4.8 分钟和 4.2 分钟，综合平均耗时为 4.7 分钟。三台终端上最大耗时为 6.5 分钟，最低耗时仅 3.5 分钟，整体波动范围较小，表明工具对终端配置差异不敏感，能够在不同性能的办公终端上稳定运行。

为更客观地对比工具与传统人工解析的性能差异，从运维班组历史统计记录中提取与测试集相同专业分布的 80 份报文的人工解析耗时数据。工具耗时按每份报文在三台终端上的平均解析耗时统计，共 80 个样本；人工耗时为同一批 80 份报文的历史人工解析记录。由于历史记录未保留逐次耗时明细，人工基线仅统计平均耗时和耗时范围，未计算中位数与标准差。人工解析与工具解析的耗时对比结果如表 4 所示。

Table 3. Parsing time under different terminal configurations
表 3. 不同终端配置下的解析耗时

测试终端	平均耗时(分钟)	最大耗时(分钟)	最小耗时(分钟)
终端 A	5.2	6.5	4.1
终端 B	4.8	5.9	3.8
终端 C	4.2	5.3	3.5
综合平均	4.7	5.9	3.8

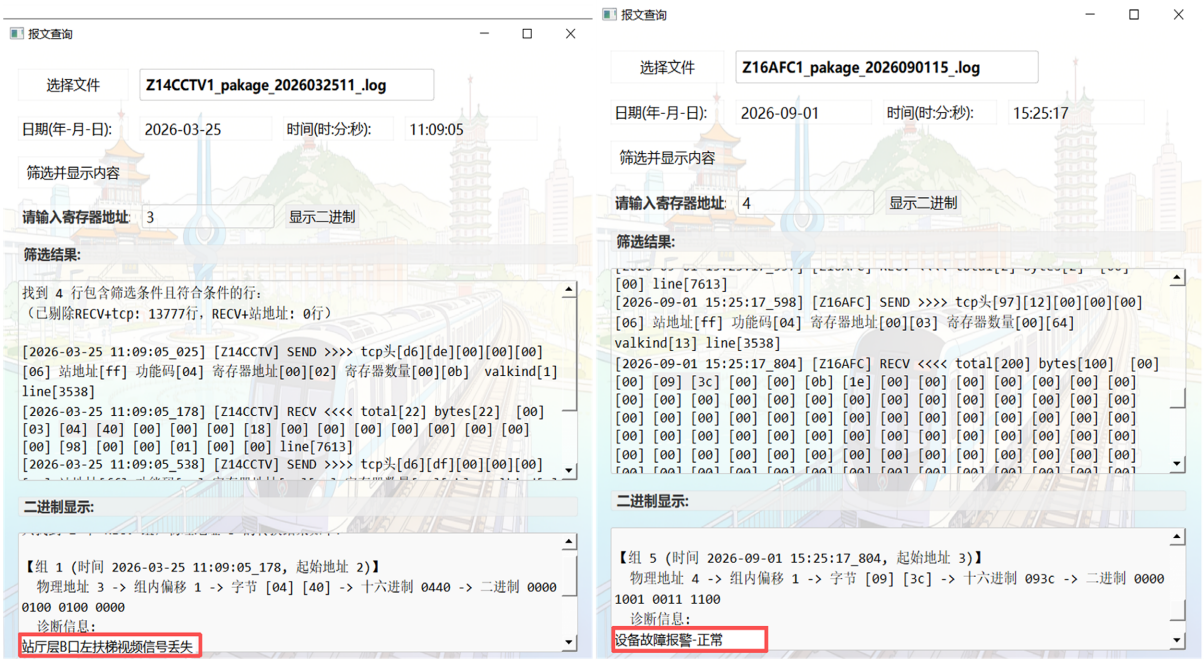
Table 4. Comparison of parsing time between manual parsing and the proposed tool
表 4. 人工解析与工具解析耗时对比

方式	样本数	平均值/min	最大值/min	最小值/min
人工	80	65	85	45
工具	80	4.7	6.5	3.5

由表 4 可知，人工解析平均耗时 65 分钟，工具解析平均耗时 4.7 分钟，效率提升约 92%。工具将单次报文解析时间压缩至 5 分钟以内，显著缩短了故障响应周期。同时，工具解析的最大耗时 6.5 分钟、最小耗时 3.5 分钟，波动范围远小于人工解析的 45~85 分钟，表明工具在显著提升效率的同时，具有更好的稳定性。

3.2.2. 解析准确率测试

将工具对 80 份报文(报文未参与规则库制定)的解析结果与专家判定的标准答案逐项比对，逐字段核对故障专业、故障设备、故障类型、寄存器状态变化量等关键信息。结果显示，80 份报文全部关键字段与标准答案完全一致，解析准确率达 100%，其中 8 个专业典型分析结果如图 7~10 所示。



(a) CCTV专业解析结果　　(b) AFC专业解析结果

Figure 7. Parsing results for CCTV and AFC subsystems
图 7. CCTV 与 AFC 专业解析结果

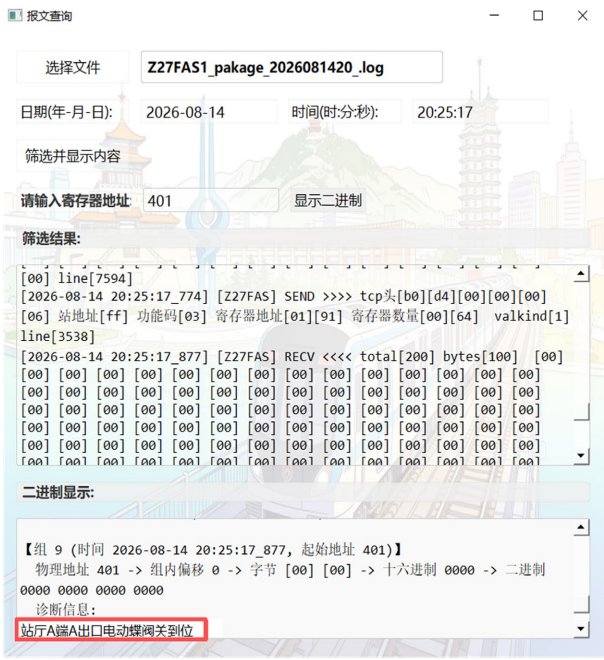


(a) BAS专业解析结果



(b) SPS专业解析结果

Figure 8. Parsing results for BAS and SPS subsystems
图 8. BAS 与 SPS 专业解析结果



(a) FAS专业解析结果



(b) PA专业解析结果

Figure 9. Parsing results for FAS and PA subsystems
图 9. FAS 与 PA 专业解析结果



(a) PIS专业解析结果　　(b) PSD专业解析结果

Figure 10. Parsing results for PIS and PSD subsystems
图 10. PIS 与 PSD 专业解析结果

在故障类型的细粒度判定上，报文可精准定位二级至三级故障子类，表明工具不仅能够识别故障所属专业与设备，还能提供细粒度的故障分类信息，为运维人员精准定位根因提供了有力辅助，有效验证了规则库与智能解析算法的可靠性与完备性。

尽管测试集准确率达 100%，未出现错误案例，但分析发现工具在以下场景仍存在潜在错误风险：一是厂商私有协议中寄存器地址偏移与标准协议不一致，可能导致寻址错误；二是同一寄存器位在不同工况下语义不同，规则库若仅按单一语义匹配可能误判；三是报文含未定义的扩展寄存器时，工具将输出“暂未定义诊断规则”，无法定位故障。上述场景在本次测试集中未出现，但需在后续规则库维护中重点关注。

3.2.3. 运行稳定性测试

工具在三台终端上连续运行，累计完成 240 次(80 份报文 ×3 台终端)解析任务，期间未出现程序报错、界面卡顿或系统异常等现象，运行稳定性良好。工具打包为单个 exe 文件，所有解析均在本地内存中完成，不产生本地缓存文件，关闭即清除，充分验证了其在基层班组标准办公终端上长期部署的可行性。

4. 工具长期维护与局限性讨论

工具上线并不意味着知识库建设的结束，而是进入持续维护阶段。由于综合监控系统涉及专业多、厂商多、协议异构，规则库和寄存器地址映射必须随现场设备、协议版本和点位表变化而持续更新。为此，本文建立了如下规则库更新流程：现场运维人员或厂商提出协议/点位变更需求后，由轨道自动化专业人员采集脱敏报文样本并标注故障特征；开发人员依据标注结果修改或新增规则，经双人复核后进行回归测试；测试通过后发布新版本规则库，并在小范围班组试运行，确认无误后全面推送；若出现异常，可回滚至上一稳定版本。

在版本管理方面，采用规则库与程序解耦、语义化版本控制和变更日志机制。每条规则记录适用专业、寄存器地址、状态位、告警等级、生效日期和审核人；规则库版本号可独立于程序版本号进行管理，并保留历史版本与基线测试集，以支持问题追溯和快速回滚。

维护成本主要包括信息采集、规则编写、测试验证、版本发布和培训等环节。初期构建覆盖多专业的规则库成本较高；后续增量维护成本与协议变更频率、新增专业数量及故障类型复杂度相关。通过模板化规则、双人复核、自动化回归测试和案例库沉淀，可有效降低长期维护成本。

当前工具仍存在一定局限：一是协议变更、点位表调整或厂商私有定义更新时，必须人工介入更新规则，存在一定滞后性；二是面对未覆盖的故障类型，可能出现漏判或无法解析；三是跨专业关联分析和根因推理能力仍有限；四是离线部署虽保障了数据安全，但版本更新需人工分发。未来可从协议模板抽象、规则半自动生成、本地化小模型辅助研判、知识图谱关联分析和增量学习等方面改进，以降低定制化适配与长期维护成本。

5. 总结

本文设计了一套轻量化、本地部署的城市轨道交通综合监控运维报文智能解析工具。工具采用“规则引擎 + 智能匹配算法”架构，构建覆盖 10 余个专业的解析规则库与寄存器地址映射数据集，实现报文专业自动判别、上下行指令区分及故障精准定位；基于 QT 封装为单个约 56 MB 可执行文件，免安装、双击即用、完全离线运行。在本文实验条件下，80 份测试报文平均解析耗时 4.7 分钟，较人工解析效率提升超 92%，准确率达 100%。

工具核心知识库具有高度定制化特征，其规则库、寄存器映射和故障样本均基于特定线路的厂商协议、点位表和历史报文构建，难以直接迁移至其他线路或厂商系统。因此，本文所述“可复制性”主要指“规则库标准化－寄存器映－规则引擎－离线封装－现场验证”的设计思想与工程路径可供借鉴，而非工具本身即插即用；跨线路应用仍需重新开展协议梳理、点位校准、专家标注和规则库重构。未来将探索协议模板抽象、规则半自动生成与知识库版本管理，以降低定制化适配与维护成本。该方案可为轨道交通行业提供一种安全可控、低成本、方法可借鉴的本地化报文解析思路。

参考文献

[1] 宋奇峰, 戚建国, 周厚盛, 等. 考虑换乘站改造条件的城市轨道交通跨线交路服务设计优化[J]. 交通运输系统工程与信息, 2026, 26(5): 142-151.

[2] 杨静俭, 杨斌. 城市轨道交通综合监控系统智能化技术研究[J]. 数字通信世界, 2025(9): 7-9.

[3] 阳德彪. 城市轨道交通综合监控报警和事件优化建议[J]. 科技资讯, 2024, 22(19): 14-17.

[4] 施德顺. 轨道交通综合监控系统调度管理应用[J]. 人民公交, 2026(16): 62-64.

[5] 刘孟栋. 城市轨道交通综合监控系统的数据安全分析[J]. 网络安全技术与应用, 2023(7): 125-126.

[6] 姚杰, 江开雄, 张松琛, 等. 城市轨道交通综合监控项目全生命期管理系统研究[J]. 绿色建造与智能建筑, 2025(11): 169-172.