한국산학기술학회논문지
Vol. 10, No. 6, pp. 1245-1255, 2009

# 철도 차량용 이벤트 레코더를 위한 분석 소프트웨어 개발

한광록[1], 장동욱[1], 김광열[2], 손석원[3*]
[1]호서대학교 컴퓨터공학과
[2]호서대학교 메카드로닉스공학과
[3]호서대학교 뉴미디어학과

# Development of Analysis Software for Railway Vehicle Event Recorder

**Kwang-Rok Han[1], Dong-Wook Jang[1], Kwang-Ryeol Kim[2] and Surgeon Sohn[3*]**
**[1]Dept of Computer Engineering, Hoseo University**
**[2]Dept of Mechatronis Engineering, Hoseo University**
**[3]Dept of New Media, Hoseo University**

**요 약** 최근 철도차량에도 객관적이고 신속 정확하게 사고 원인을 분석하고, 사고를 미리 예방하기 위하여 여러 나라에서 철도 차량의 운행정보가 기록되는 블랙박스 즉, 이벤트 레코더의 설치를 법제화하고 있고, 이에 대한 연구를 진행하고 있다. 또한 사고에 대한 정확한 판단을 위하여 저장된 데이터를 분석하고 표현할 수 있는 분석 소프트웨어가 요구된다. 따라서 본 논문에서는 이벤트 레코더에 기록된 데이터를 분석하고 음성과 영상을 재생하는 분석 소프트웨어를 개발하였다. 본 논문의 이벤트 레코더 분석 소프트웨어는 신속하고 정확하게 사고 원인을 규명할 수 있고, 구간별 운행 패턴과 기관사의 습관 등을 파악할 수 있다. 또한 이미 발생한 사고 상황에 대해 영상과 음성을 함께 분석함으로써, 차후 발생할 수 있는 사고를 미연에 방지할 수 있을 것으로 기대한다.

**Abstract** Recently, to analyze the cause of the railway accident objectively and quickly and prevent the accident, many countries are legislating for the installation of the black box what we call an event recorder, which records information about the operation of railway vehicle. Thus, the study of the event recorder has been in progress. Moreover, the analysis software that can analyze and express the stored data in the event recorder is required for the correct decision on the accident. Therefore, in this paper, we presented a design of analysis software which analyzes the data, plays the audio and video in the event recorder system. This software can quickly and accurately identify the cause of the accident and recognize the driving patterns and habits of the driver according to the operating section. In addition, by analyzing the audio and video data simultaneously in the previous accident, we expect that it is possible to prevent accidents in advance.

**Key Words :** Event Recorder, Analysis Software, Simulation, User Interface View

## 1. 서론

일반적으로 블랙박스란 항공기 운항 중 발생하는 데이터를 기록하는 장치를 말하며 항공기 사고 원인을 규명하는데 없어서는 안 될 장비가 되었다. 최근에 철도의 속도가 고속화, 자동화됨에 따라 철도 사고 역시 항공기 사고처럼 대형 인명사고로 나타나고 있다. 몇 가지 사고 사례를 보면, 1998년 독일 ICE가 하노버 인근 에셰데에서 바퀴의 균열로 인한 탈선으로 열차가 인근 교각과 충돌하면서 101명이 사망하고, 105명이 부상하는 사고가 발생하였다. 터키에서는 2004년 7월 터키 북서부 사카랴주에서 새로 도입한 고속열차가 탈선해 36명이 사망하고

*교신저자: 손석원(sohn@hoseo.edu)
접수일 09년 01월 30일 수정일 (1차 09년 03월 24일, 2차 09년 06월 15일) 게재확정일 09년 06월 17일

79명 이상 이 부상했다. 이 사고는 이스탄불부터 앙카라까지의 노선으로 고속열차가 개통 한 달 만에 기존의 노후한 철로를 그대로 이용하다가 곡선부위에서 발생하였다[1,2].

이렇듯 최근 철도 사고는 인명사고가 발생하면 대부분 대형사고로 이어짐에 따라 세계 각국에서는 객관적이고 신속, 정확한 사고 원인분석과 사고예방을 위한 기관사 운행관련 교육 등을 위해 항공기의 블랙박스와 같은 장비, 즉 이벤트 레코더(Event Recorder)의 설치 의무화와 같은 철도차량 안전에 관한 법률이 제정되고 관련 단체가 창설되기 시작했다. 국내의 경우 관련 법규로는 2004년 10월 처음으로 철도 안전법이 법률로 제정되었으며 철도 안전법 시행령, 철도 안전법 시행규칙을 거쳐 2006년 2월 철도안전종합계획에 열차속도 및 운행 기록계 설치 의무화와 철도사고 통계분석 프로그램 개발이 추진되었고, 2007년 6월 열차충돌·화재 등 각종 철도사고에 대비하여 기록 장치를 안전하게 보존할 수 있도록 설치기준 등 세부 지침을 마련하여 시행 되었다[3]. 유럽은 유럽 전반에 걸친 열차제어시스템의 효율적인 운용을 위해 1995년 7월 프랑스, 독일, 이탈리아 간에 유럽 경제이익단체(GEIE)를 창설하였다. 이 단체의 목적은 국제철도연맹, 유럽 연합과 함께 유럽의 서로 다른 철도망 전반과의 인터페이스를 위해 봉사하는 것은 물론 상호운용성의 문제점을 실제로 실현하는데 있어서 여러 철도 관련 전문 산업체를 적절하게 운영하는 것에 있다. 한편 유럽 연합은 국제철도연맹의 지원과 함께 유럽 열차제어 시스템의 기본 골격을 구성하는데, 이를 ERTMS(European Railway Traffic Management System)라 명명하고, 하부구조로서 열차운행 관련 안전기능을 수행하도록 주어진 프로젝트를 유럽열차제어시스템(ETCS)으로 정의하였다. 이에 대한 사양은 국제철도연맹의 유럽철도연구원 소위원회(ERRI A200)에 의뢰되었으며, 이의 개발을 위해 유럽 철도망은 하나로 통합되었다[4].

이벤트 레코더에 대한 법제화가 이루어지고 관련 단체들이 등장함에 따라 사고 발생 시 이벤트 레코더에 저장된 데이터를 토대로 보다 빠르고 정확하게 사고 원인을 규명하는데 도움을 줄 수 있는 분석 소프트웨어의 중요성도 높아지고 있다.

본 논문에서는 철도 차량 운행 시 이벤트 레코더에 기록된 데이터를 분석하는 소프트웨어의 개발에 관하여 기술한다. 분석 소프트웨어는 이벤트 레코더에 기록된 원시 데이터를 다운로드 분석/저장한 후 , 다양한 뷰어(Viwer)를 통해 분석 소프트웨어의 기본 기능인 철도차량의 상태와 운행정보를 확인할 수 있다. 그 외에 차량 내의 음성과 영상을 같이 제공함으로써 기록될 당시의 상황에 대해 더욱 정확하고, 효과적인 분석이 가능하도록 한다. 또한 기록된 데이터를 바탕으로 기록당시 철도차량의 운전실 상황을 한눈에 파악할 수 있도록 시뮬레이션(Simulation) 기능도 구현한다.

## 2. 관련 연구

이벤트 레코더는 철도 차량의 운행 상태에 관한 속도, 센서 및 제어 정보, 통신 등 각종 정보를 실시간 수집·저장하고, 열차의 기능을 감시하며, 열차 사고 시 원인을 정확하고 빠르게 파악하여, 동일한 사고를 예방하고 나아가 신호설비의 유지보수 및 고장 분석 등을 통하여 발생할 수 있는 사고를 미연에 방지하기 위해 사용되는 중요한 장치이다[1,2,5,6].

IEEE에서 정의한 이벤트 레코더는 다음의 3가지 요소를 가진다[7,8].

- 기능, 운영 및 전기적 특성 - 기능의 중요한 측면은 이벤트 레코더 자체의 상태를 확인하고 알리는데 있다. 운영 및 전기적 특성으로는 활동 및 데이터 발생에 대한 모니터링 입력 채널은 기준이 필요하다. 내장 배터리는 이벤트 레코더의 필수적 요소이다. 수집된 데이터를 다운로드하는 프로토콜은 제한이 없으며 가능한 프로토콜을 사용한다.
- 필수 입력 - 이벤트 레코더에 필수로 입력되는 신호들인 날짜, 시스템 시간, 브레이킹 명령, 속도 센서, 출입문 개폐 상태 등의 신호는 IEEE의 Working Group의 요구사항을 따른다.
- 충격 생존 특성(사고 시 장비 내구성) - 열차 사고 시 이벤트 레코더가 화재, 충돌, 화학물질 등에 얼마나 견뎌야 하는 지를 나타낸다.

ERTMS/ETCS Class1을 만족하는 문서(Form Fit Function Interface Specification for the Juridical Recorder-Downloading Tool)는 이벤트 레코더와 분석 소프트웨어간의 프로토콜, 프로토콜에 쓰이는 메시지의 종류, 메시지 정의, 메시지 구조 등이 기술되어 있다[9].

이벤트 레코더의 한 종류인 ATESS(속도기록계장치)는 현재 국내의 고속열차 KTX에 장착되어 운영중이다. ATESS는 열차속도 측정, 열차속도 및 주요정보의 기록, VACMA(기관사 경계장치), TSL(열차속도제한장치)등의 주요 기능을 제공한다. 또한 ATESS는 RU(Recorder Unit)와 JRU(Juridical Recorder Unit)의 두 개의 장치로 구성되어 있다. RU는 국내 및 프랑스 표준을 만족하며 열차의 운전 데이터를 기록하는 장치이며, JRU는 ATP(Automatic Train Protection) 설치차량의 개량 방안

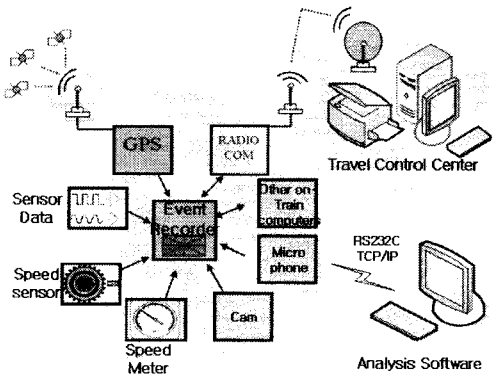
ERTMS 표준을 만족하며 MVB링크에서 수신되는 데이터를 기록하는 장치이다[10].

이벤트 레코더 분석 소프트웨어로는 프랑스의 Faiveley사의 SAM(Software for Analysis and Maintenance)을 들 수 있다. Java2 Platform 및 Enterprise Edition(J2EETM) application으로 데이터를 그래픽, 텍스트, 16진법으로 표시하며 화면에 표시되는 데이터는 서로 동기화된 데이터를 보여준다. 아날로그 그래프는 4개, 디지털 그래프는 10개 까지 지원하며 4 개의 그래프 비교 축을 제공하고 그래프의 수평, 수직 확대를 제공한다. 또한 필터링과 엑셀파일로 Data Export도 제공한다[11].

## 3. 이벤트 레코더 분석 소프트웨어 설계

### 3.1 이벤트 레코더의 개요

이벤트 레코더는 그림 1과 같이 구성된다. GPS, 열차 정보, 센서 데이터(문 개폐, 속도, 정지거리 등), 운전자의 운전반 조작 행위, 운전사령실(Travel Control Center)과의 통신 내용, 열차 내 영상, 음성 등의 정보를 저장하며 분석 소프트웨어와 통신을 통해 데이터를 전송한다[12].




[그림 1] 이벤트레코더의 구성

### 3.2 분석 소프트웨어 설계

분석 소프트웨어(ASER)는 관련 산업계의 다음의 요구에 따라 설계하였다.

용이성 - 빠른 분석을 위해 그래프, 데이터 테이블, 메시지 뷰어와 같은 다양한 뷰어를 구성 하였고, 각각의 뷰어가 시간을 기준으로 싱크가 이뤄지도록 하여, 직관적인 분석이 가능하도록 설계 하였다.

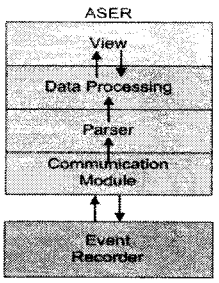
정확성 - 시간의 흐름 순서대로 열차 속도 및 각종 센서의 정보를 시뮬레이션 할 수 있을 뿐만 아니라, 이벤트 당시 열차 내의 CCTV와 같은 비디오 정보, 방송이나 통신 내용과 같은 음성 정보를 함께 확인할 수 있어, 당시 열차의 상황을 정확하고 빠르게 파악할 수 있도록 설계 하였다.

범용성 - 차후 이벤트레코더와 분석소프트웨어간의 범용성을 위하여 현재 유럽연합에서 사용중인 프로토콜을 최대한 참고하여 설계함으로서 분석소프트웨어간의 범용성으로 하였으며, 또한 분석 데이터를 Excel 데이터로 추출이 가능하여 다양한 목적으로 활용할 수 있다.

편리성 - 이벤트 레코더와의 편리성을 위하여 RS-232C를 비롯한 TCP/IP와 외장 디스크등을 이용한 데이터 전송이 가능하도록 설계하여 온/오프라인 환경과 유/무선 환경에 대응할 수 있도록 설계하였다.

데이터의 안전성 - 분석된 원시데이터는 가공되지 않은 원본 상태로 인덱스를 추가 생성하여 저장되며, 인덱스를 이용하여 원시데이터와 분석된 데이터를 즉시 비교할 수 있도록 설계하여 데이터가 정확함을 확인 할 수 있다.

산업계의 요구에 따라 설계된 ASER의 의 구조는 그림 2와 같다. 하드웨어인 이벤트 레코더로부터 각종 센서 신호의 데이터를 다운로드하는 통신 인터페이스, 그리고 획득한 원시 데이터를 파싱하여 우리가 필요한 데이터로 가공하는 파서(Parser), 가공된 데이터를 저장하는 데이터베이스, 그리고 사용자에게 이벤트 레코더의 데이터를 보기 쉽게 표현하는 View로 구성된다.



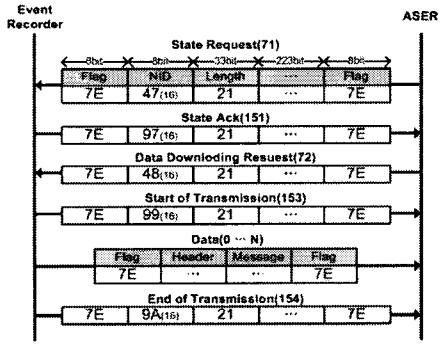

[그림 2] ASER의 구조

ASER의 UI는 Visual Basic을 이용하였으며, 내부 인터페이스인 통신, 파서, 데이터처리 부분은 MFC를 이용하여 DLL로 개발하였다. 많은 양의 데이터를 저장하기 위해 데이터베이스를 이용하였으며, MSSQL, MySQL, ACCESS를 모두 지원하며, ASER에서 필요시 선택할 수 있도록 개발 하였다.

### 3.2.1 통신 인터페이스

이벤트 레코더에 기록된 데이터는 차량운행이 종료된 후 RS232C, TCP/IP 프로토콜 또는 USB를 이용하여 원시 데이터를 ASER로 다운로드 한다.

프로토콜과 메시지는 유럽연합의 ERTMS/ETCS - Class 1 Subset 027 문서[7]을 참조하여 설계하였으며 그림 3은 이 과정을 정의한 프로토콜을 나타낸 것이다.




[그림 3] 통신 프로토콜

프로토콜은 다음과 같이 정의된다.

① 먼저 ASER에서 이벤트 레코더가 전송 가능한 상태인지를 확인하는 Start Request(71) 메시지를 전송한다.

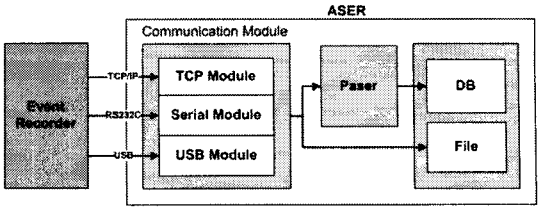
② 이벤트 레코더는 ①의 응답으로 State Ack(151) 메시지를 전송한다.

③ ASER는 ②의 메시지를 수신하면, 원시 데이터의 전송을 요청하는 Dat Download Request(72) 메시지를 이벤트 레코더로 전송한다.

④ 이벤트 레코더는 ③의 메시지를 수신하면, 원시 데이터의 전송이 시작 알리는 Start of Transmission (153) 메시지를 ASER로 전송하고, 이어 원시 데이터를 전송한다.

⑤ 이벤트 레코더가 원시 데이터를 모두 전송을 완료하면, 데이터 전송 완료를 알리는 End of Transmission (154) 메시지를 전송한다.

통신 인터페이스는 앞서 정의한 프로토콜을 사용하며 그림 4와 같이 크게 3가지 모듈로 구성되어 있다. 온라인의 TCP/IP 모듈과 RS232C 모듈 그리고 오프라인의 USB 모듈이 있다. USB 모듈 은 TCP/IP와 RS232C 통신을 이용하여 원시 데이터를 전송하기가 용이하지 않을 경우에 직접 USB를 이용하여 원시 데이터를 ASER로 복사한다.




[그림 4] ASER의 통신인터페이스

[표 1] 메시지 정의

| Message No. | Message description |
|---|---|
| 0 | Intentionally removed |
| 1 | General message |
| 2 | Data entry completed |
| 3 | Emergency brake order |
| 4 | Service brake order |
| 5 | events |
| 6 | Telegram form balise |
| 7 | Message form euroloop |
| ... | ... |
| 28 | RTD |
| 29 | Audio |
| 30 | Video |
| 31-150 | Spare |
| 151 | State ack |
| 152 | Download failure |
| 153 | Start of Transmission |
| 154 | End of Transmission |
| 155-255 | Spare |

※ 0~150번은 Data Message 151~255는 Control Message

표 1은 프로토콜에서 사용하는 메시지를 나타낸 것이다. 메시지는 크게 두 가지 종류가 있으며 0번부터 150번 메시지는 데이터를 위한 메시지이고 151번부터 255번까지는 통신 제어를 위한 메시지이다. 데이터를 위한 메시지에는 열차 속도, 브레이크 상태, 에러 및 장애 메시지, 디스플레이 메시지, 목표 속도, 제한 속도 등 열차 운행에 관계된 모든 메시지가 포함되어 있다[7].

본 논문의 이벤트 레코더는 비행기의 블랙박스와 같이 철도 차량 내의 음성과 영상을 저장하는 기능을 구현하였다.

이를 위해 ERTMS/ETCS - Class 1 Subset 027[7]의 프로토콜에 오디오 메시지(29번 메시지), 비디오 메시지(30

번 메시지)를 추가하였다. 표 2와 표 3은 추가된 오디오와 비디오 메시지의 구조를 나타낸다.

**[표 2]** 오디오 메시지의 구조

| Variable | Length(Bit) | Description |
|---|---|---|
| BPS | 2 | 비트 전송률 00 : 16 (Kbps) 01 : 32 10 : 64 11 : 128 |
| SAMPLE SIZE | 2 | 오디오 샘플 크기 00 : 4 (Bit) 01 : 8 10 : 16 11 : 32 |
| CHANNEL | 1 | 채널 0 : mono 1 : stereo |
| SAMPLE SPEED | 2 | 오디오 샘플 속도 00 : 12 (KHz) 01 : 16 10 : 22 11 : 44 |
| SIZE | 33 | 오디오 파일 크기 |

**[표 3]** 비디오 메시지의 구조

| Variable | Length(Bit) | Description |
|---|---|---|
| XSIZE | 10 | 이미지 너비 (Pixel) |
| YSIZE | 10 | 이미지 높이 (Pixel) |
| LTIME | 20 | 총 재생시간 (초) |
| BPS | 2 | 비트 전송률 00 : 16 (Kbps) 01 : 32 10 : 64 11 : 128 |
| CODEC | 3 | 코덱 종류 000 : mpeg4 001 : wmv 010 : divx 011 : spare |
| FSPEED | 2 | 프레임 속도 00 : 16 (프레임/초) 01 : 32 10 : 64 11 : 128 |
| DSPEED | 2 | 데이터 속도 00 : 1103 (Kbps) 01 : 1410 10 : spare |
| SIZE | 33 | 비디오파일 크기 (Byte) |

오디오 및 비디오 메시지를 녹음, 녹화하는 기준은 표 4와 같다. 각 항목에 해당하는 이벤트가 발생 시 센서가 이를 감지하여 이벤트 레코더에 기록하게 된다. 예를 들어 100dB 이상의 고음이 감지될 때 오디오와 비디오 메시지 모두 기록하고 비상브레이크 작동 시에는 비디오만 기록하게 된다.

**[표 4]** 녹음 및 녹화 기준

| 항 목 | 오디오 | 비디오 |
|---|---|---|
| 고음(100dB 이상) | ○ | ○ |
| 비상 벨 | ○ | |
| 운전 사령실 호출 | ○ | |
| 안내/비상 방송 | ○ | |
| 충돌(1kN 이상) | ○ | ○ |
| 비상 브레이크 | | ○ |
| 문 개 · 폐시 | | ○ |
| 교각, 터널 진입 시 | | ○ |
| 역사 진입 시 | | ○ |

### 3.2.2 파서(Parser)

파서는 이벤트 레코더에서 전송된 원시 데이터를 분석을 한다. 이벤트 레코더에서 저장한 각종 센서 등으로부터 들어오는 데이터는 저장공간의 효율적인 활용을 위해 비트로 데이터를 저장하고, 분석 소프트웨어는 이를 쉽게 추출하기위해 비트 필드 연산을 이용한다. 비트 필드는 비트 단위의 데이터를 처리하기 위한 구조체로서 그림 5와 같은 문법 구조를 갖는다.

```
struct bitExData{
    int x : 4; // [데이터 형] [항목의 이름] : [비트의 수]
    int y : 4;
  };
```

**[그림 5]** 비트 필드의 문법

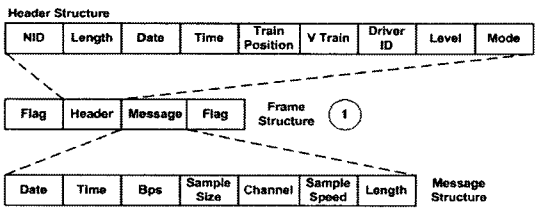
그림 6은 이 비트 필드를 이용하여 작성한 오디오 메시지의 구조체 이다.

```
struct M29{
        unsigned int YEAR       : 7;
        unsigned int MONTH      : 4;
        unsigned int DAY        : 5;
        unsigned int HOUR       : 5;
        unsigned int MINUTES
        : 6;
        unsigned int SECONDS
        : 6;
        unsigned int TTS        : 5;
        unsigned int BPS        : 2;
        unsigned int SAMPLESIZE
  : 2;
        unsigned int CHANNEL
        : 1;
        unsigned int SAMPLESPEED    : 2;
        unsigned __int          64 LENGTH; // 오디
오파일 크기(33bit)
  };
```

**[그림 6]** 오디오 메시지의 구조체

이벤트 레코더에 저장되는 프레임 구조는 그림 7의 ①과 같다.

프레임 구조는 각각의 메시지 구분을 위한 플래그와 메시지 번호(NID), 메시지 길이, 메시지 생성날짜 등의 데이터로 구성된 헤더와 해당 메시지의 실제 데이터가 들어있는 메시지로 구성된다.




**[그림 7]** 오디오 메시지의 프레임 구조

데이터 분석 시 헤더는 헤더 구조체에 넣고 메시지는 헤더에 있는 메시지 번호 값을 참조하여 그 번호에 해당하는 메시지 구조체에 넣게 된다. 구조체에 넣은 메시지 데이터는 데이터베이스에 추가된다. 오디오 메시지와 영상 메시지는 데이터 분석 시 음성 및 영상 파일을 추출하여 정해진 저장장소에 저장한다.

### 3.2.3 데이터베이스

이벤트 레코더에서 전송된 원시 데이터는 파일 형태이기 때문에 이를 직접 사용할 경우 매번 분석을 할 때마다 파싱을 해야 하므로 효율적이지 못하다. 또한 원시 데이터가 많아질수록 관리해야 할 원시 파일도 많아지므로 관리를 위해 별도의 기법이 필요하기 때문에, 또한 효율적이지 못하다.

따라서 원시 데이터는 미리 분석을 위해 파싱을 해야 할 필요성이 있으며, 파싱된 데이터를 보관하는 방법으로는 XML과 같은 정형화된 파일로 보관하는 방법과 데이터베이스를 이용하는 방법이 있다.

정형화된 파일로 보관하는 방법은 매 운행 시 마다 이벤트 레코더가 원시 데이터를 생성하는데, 이때마다 각각의 파싱된 파일이 만들어진다. 이 역시 파일을 관리하기 위한 별도의 기법의 필요하다.

데이터베이스를 이용할 경우 운행 횟수가 많아지더라도 파싱된 파일이 생기지 않기 때문에 정형화된 파일로 보관하는 방법보다 효율적으로 관리할 수 있다.

ASER는 데이터베이스를 이용함으로서 데이터를 효율적으로 관리할 수 있다.

### 3.2.4 사용자 뷰(User Interface View)

(1) 뷰의 구성

사용자 뷰는 GraphView, GridView, HexView, MessageView의 4가지로 구성되며 그림 8은 뷰의 구성도를 나타낸다.

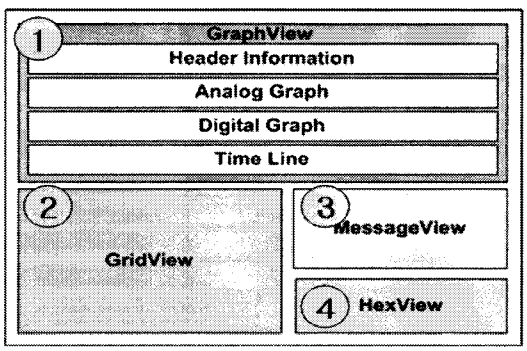
① GraphView : GraphView는 4개의 영역으로 나눈다.
·Header Information : 헤더 정보를 표시하는 영역
·Analog Graph : 속도와 출력 등의 아날로그 값을 가진 데이터의 그래프를 그리는 영역
·Digital Graph : 도어 오픈 상태, 브레이크 작동유무 등의 디지털 값을 가진 데이터의 그래프를 그리는 영역
·Time Line : 시간정보와 해당 시간대 오디오 및 비디오 메시지의 유무를 나타내 주는 영역

GraphView에서 그래프를 그리는 과정은 다음과 같다. 먼저 그래프가 그려질 대상인 화면의 크기를 지정한 후, 전체 레코드 수를 스크린이 가질 수 있는 최대값과 비교하여 스크린 범위를 넘지 않게 조절한다. 그리고자하는 그래프의 개수를 파악한 뒤 그래프 출력 배율에 따라 한 화면에 보여줄 수 있는 데이터를 X축 값으로 설정하고, 구간의 최소값과 최대값을 Y축 값으로 설정 한 후, 그래프를 그린다.

② GridView : GridView는 행과 열로 구성되며 행은 시간대별 샘플링 데이터를 의미하며 열은 변수들을 의미한다.

③ MessageView : GridView, GraphView, HexView에서 데이터 선택 시 선택한 데이터에 해당하는 변수의 의미와 값을 출력한다.

④ HexView : HexView는 원시데이터를 hex값으로 보여주기 위한 것으로 다른 3개의 뷰 들과는 달리 원시파일에서 데이터를 읽어오며 16진 데이터의 연속된 행으로 표시한다.



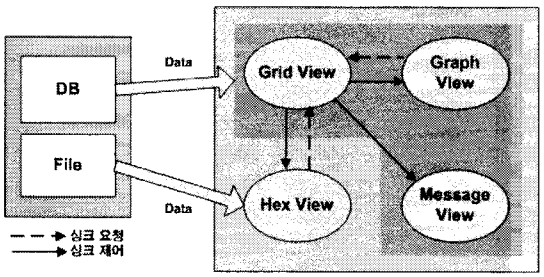

**[그림 8]** 뷰의 구성

(2) 뷰 동기 제어

가) 뷰 동기화

뷰 실행 시 GridView는 선택된 구간의 모든 데이터를 바인딩하며, GraphView는 선택된 구간 중 그래프를 그리는데 필요한 데이터만 바인딩하고, MessageView는 싱크 요청 시에만 데이터베이스에서 데이터를 바인딩하며 HexView는 원시 데이터 파일에서 선택된 구간의 모든 데이터를 읽어와 표시한다. 싱크 요청은 MessageView를 제외한 GridView, GraphView, HexView에서 요청 할 수 있으며 싱크 제어는 GridView만 가능하다.

그림 9는 각 뷰의 싱크 요청 및 제어를 나타낸 것이며 각 뷰의 싱크 요청 및 제어 과정은 다음과 같다.




[그림 9] 뷰의 싱크 및 제어도

① GraphView에서 동기화 하고자 하는 부분의 그래프 좌표를 식(1)에 의해 GridView의 키 값으로 변환하여 GridView에 싱크요청을 한다.

$$(a+b-c)*d \quad (1)$$

a : 그래프 시작 값
b : 현재 마우스의 X좌표
c : 그래프 시작 지점의 X좌표
d : 배율

GraphView로부터 싱크 요청을 받은 GridView는 키 값을 통해 선택한 지점의 인덱스값을 확인 후 GraphView, MessageView, HexView를 인덱스 값 기준으로 출력하게 함으로써 동기화 하게 된다.

② GridView에서 동기화 하고자 하는 데이터 행을 선택하면 바로 인덱스 값을 확인 하여 싱크 요청을 한다. 싱크 제어 과정은 ①의 싱크 제어 과정과 같다.

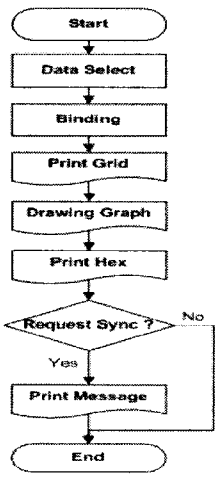
③ MessageView는 싱크 요청 과정은 없고 싱크 제어 과정만 있으며 싱크 제어 과정은 ①의 싱크 제어 과정과 같다.

④ HexView는 원시 데이터 파일에서 데이터를 읽어옴으로써 동기화를 위해 데이터베이스에 저장한 해당 메시지의 파일위치 데이터를 사용한다. 동기화하고자 하는 행을 선택하면 선택한 행이 속한 메시지 시작 지점의 위치 값을 구해 싱크 요청을 한다. 이때 구한 위치 값을 파일위치 데이터와 비교하여 선택한 행의 인덱스 값을 확인하며 싱크 제어 과정은 ①의 싱크 제어 과정과 같다.

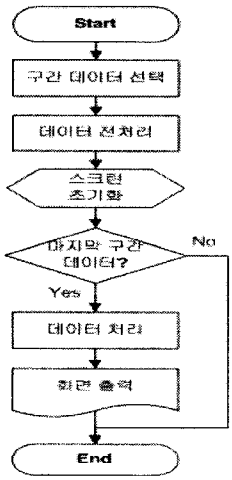
나) 뷰의 출력흐름

뷰의 출력흐름은 그림 10과 같다. 출력할 구간을 선택하면 데이터베이스로부터 그리드에 데이터를 바인딩 한 후 차례로 그리드, 그래프, 원시 데이터를 출력한다. 그 후 싱크 요청이 있었는지를 확인 후 메시지를 출력한다.

싱크를 요청한 경우와 하지 않은 경우 출력되는 데이터의 차이점은 MessageView의 데이터 표시 여부이다. 싱크를 요청한 경우에만 MessageView에 데이터를 표시하기 때문이다. 또한 GraphView에 싱크를 요청한 데이터에 대해 세로축으로 선이 표시되는 점이 다르다.




[그림 10]
뷰의 출력흐름




[그림 11]
시뮬레이션의 흐름

### 3.3 시뮬레이션

시뮬레이션이란 선택된 구간을 실제 열차가 운행한 것과 같게 각종 계기를 시간에 맞게 동기화시켜 재현 되도록 구현한 기능이다. 시뮬레이션은 선택된 구간의 데이터를 바탕으로 그래픽 요소를 이용하여 사용자가 알아보기 쉽게 표현하였다. 선택한 데이터는 실제 열차가 운행하면서 기록한 데이터이며 운행당시 기관실의 각종 계기데이터를 재현하여 보여 준다.

그림 11은 시뮬레이션의 흐름을 나타낸다. 시뮬레이션도 뷰 모드와 마찬가지로 구간선택 과정을 거쳐 시뮬레이션 할 구간을 선택하게 된다. 구간이 정해지면 데이터

베이스로부터 읽어온 데이터를 전처리 과정을 통해 시뮬레이션될 데이터를 처리하고, 화면에 출력할 준비를 한다. 출력할 준비를 마쳤으면 시간의 흐름에 따라 준비된 데이터를 출력을 한다.

## 4. 실험 및 평가

### 4.1 통신 실험

열차 운행에 관련되어 지속적으로 저장되는 운행 데이터는 메시지 사이즈를 33Byte라고 하고 1초당 100개의 메시지가 발생 되며 열차가 하루 평균 4시간씩 3회를 운행하며 저장할 경우 필요한 메모리 용량은 아래와 같다.

33byte × 100개 × 60분 × 60초 × 4시간 × 3회 = 135Mbyte

오디오 및 비디오 메시지 데이터의 경우에는 항상 녹음 및 녹화되는 것이 아니라 표4의 항목에 의해 저장됨으로 열차 1회 운행 시 총 녹음 및 녹화 시간은 약 30분이라 가정하며 각각 20개의 마이크로폰과 카메라가 장착되어 있는 열차가 하루 평균 4시간씩 3회의 운행을 하며 녹음 및 녹화할 경우 필요한 용량은 아래와 같다.

오디오 메시지는 mp3 코덱을 사용하며 64Kbps(비트율)를 사용한다. 비디오 메시지는 320x240의 화면 크기를 사용하며 wmv 코덱을 사용하고 초당 15프레임을 저장할 수 있는 150Kbps(전송률)를 사용한다.

오디오 메시지 : 64Kbps × 20개 × 30분 × 60초 × 3회 / 8bit = 0.86 Gbyte

비디오 메시지 : 150Kbps × 20개 × 30분 × 60초 × 3회 / 8bit = 2.03 Gbyte

이 원시 데이터를 이벤트 레코더로부터 ASER로 다운로드 받기 위해 19.2Kbps를 사용하는 RS232C와 100Mbps의 TCP/IP를 사용하여 다운로드 할 경우 걸리는 시간을 계산하면 표 5와 같다.

**[표 5]** 통신 종류에 따른 소요시간

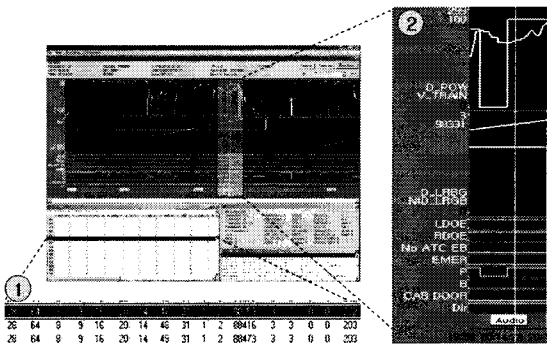
| | 운행 데이터 (135 Mbyte) | 운행 데이터 + 비디오 메시지 (2.16 Gbyte) | 운행 데이터 + 오디오 메시지 + 비디오 메시지 (3.02 Gbyte) |
|---|---|---|---|
| RS232C | 16.5시간 | 250.9시간 | 358.9시간 |
| TCP/IP | 10.9초 | 2.8분 | 3.9분 |

표 5의 소요 시간은 ASER 구동 시마다 소요되는 시간이 아니라 최초 다운로드할 경우의 시간이고 이후에는 분석 소프트웨어가 탑재된 랩탑의 데이터베이스에 저장되어 별다른 통신 과정이 필요 없게 된다.

### 4.2 동기화 실험

그림 12는 앞의 설계에 따라 구현된 ASER의 뷰 기능이다. 실험에 사용한 데이터는 2008년 9월 16일 19시부터 23시까지 운행한 열차의 데이터를 이용하였고 이 데이터를 28번 메시지로 정의하여 사용하였다.

①은 ASER에서 동기화 시켰을 때 GridView의 메시지 내용을 확대한 그림이다.




**[그림 12]** ASER의 뷰 기능

GridView에 나타난 메시지의 주요 내용은 [표 6]과 같다.

**[표 6]** GridView의 데이터

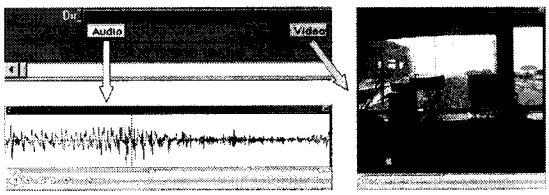
| Data (Graph의 제목) | Value |
|---|---|
| NID Message | 28 |
| Message Length | 64 |
| Year | 8 |
| Month | 9 |
| Day | 16 |
| Hour | 20 |
| Minute | 14 |
| Second | 47 |
| Station Number (NID_LRBG) | 2 |
| Distance (D_LRBG) | 88,359 |
| Train Speed (V_Train) | 202 |

표 6은 28번 메시지이고 길이는 64byte이며 2008년 09월 16일 20:14:47에 생성된 데이터라는 것을 알 수 있다. 또한 2번역을 통과하여 88,359m를 운행했으며 열차의 속도는 202Km/h이다.

②는 GridView에서 나타난 내용을 보다 쉽게 알아 보기 위해 그래프형태로 나타낸 것이며 세로축 하얀 선은

현재 동기화된 데이터를 나타낸다. V_TRAIN 선은 속도를 나타내며 최대값은 299Km/h이고 현재 속도는 표 6에 나와 있는 202Km/h임을 알 수 있다. D_POW 선은 가속 및 제동 정보를 나타내는데 가속도인지 제동정보인지는 그래프 하단에 있는 'P', 'B'의 값을 보고 알 수 있다. 현재 'P'의 값이 1이므로 D_POW 선은 가속상태임을 나타낸다. D_LRBG 선은 열차가 바로 앞의 역을 통과하여 주행한 거리를 나타내며 현재 값은 표6의 88,359m를 나타냄을 알 수 있다. NID_LRBG 선은 열차가 가장 최근에 통과한 역의 번호를 나타내므로 현재 열차는 2번역을 통과했다는 것을 나타낸다.

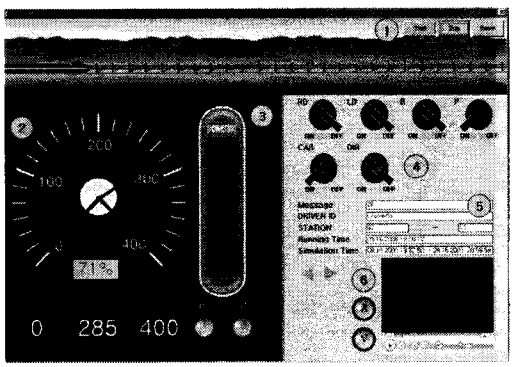
그림 13은 GraphView의 하단에 위치한 타임라인 부분에 위치한 오디오 버튼과 비디오 버튼을 클릭했을 때 실행되는 화면을 나타낸 것이다. 그림 13에서 오디오는 표 4의 안내 방송에 해당하는 데이터 이며 비디오는 표 4의 역사 진입 시 기관실의 녹화 데이터 이다.




[그림 13] 음성 및 영상 메시지 구현

### 4.3 시뮬레이션 실험

그림 14는 구현된 시뮬레이션 기능을 실행한 화면이다. 최대한 실제 기관실처럼 구성하여, 표기하도록 구현하였다.




[그림 14] 시뮬레이션 화면

①은 시뮬레이션의 Start, Stop, Reset을 할 수 있는 버튼들이다.

②의 정보는 헤더에 저장된 속도 정보를 기반으로 현재 열차의 속도를 표시한다. 이때 MAX는 현재 열차의 최대 속력으로 표시하고 이에 맞게 조정하여 표시한다.

③은 속도 및 제동 정보를 표시하는 영역이다. 이 정보를 이전 역과 다음역의 역간 거리를 확인하고, 현재 열차가 어느 위치에 있는지를 확인한다.

④의 정보는 디지털 정보를 표현하기 위한 스위치 부분으로서, 현재 브레이크 및 도어 상태를 표시하는 영역이다.

⑤의 정보는 표시되는 메시지와 각종 텍스트 정보 등을 표시하고,

⑥은 현재 시뮬레이션 되고 있는 구간에 영상이나 음성 정보가 있으면 이를 재생해주는 영역이다.

### 4.4 비교 및 평가

표 7은 현재 사용되고 있는 이벤트 레코더용 분석 소프트웨어와 본 논문에서 설계한 ASER를 비교한 표이다.

[표 7] 분석 소프트웨어간의 비교

| 항 목 | Faiveley SAM | EKE TIP | ASER |
|---|---|---|---|
| 그래프의 비교 축의 수 | 4 | | 1 |
| 아날로그 그래프의 수 | 4 | | 5 |
| 디지털 그래프의 수 | 10 | | Unlimited |
| 그래프의 수평 확대 | ○ | | ○ |
| 그래프의 수직 확대 | ○ | | |
| 그래프별 확대 | | | ○ |
| 지점간 시간/거리표기 | ○ | | |
| 오디오/비디오 재생 | | | ○ |
| 시뮬레이션 | | | ○ |
| 실시간 모니터링 | | ○ | |
| 필터링 | ○ | | |
| Data Export | ○ | | ○ |

ASER 포함 총 3가지 제품을 비교하였으나 EKE사의 TIP는 자료가 많지 않아 직접 비교하기는 어렵다. Faiveley사의 SAM은 그래프의 수평, 수직 확대 모두 지원하나 ASER는 수평 확대만을 지원 하는 대신 각각의 그래프별로 확대하는 기능을 지원한다. 또한 ASER는 오디오와 비디오 데이터의 재생이 가능하며 일정 구간에 대해 실제 열차가 운행한 것과 같게 각종 계기를 시간에 맞게 동기화시켜 재현 되도록 구현한 시뮬레이션도 지원한다. 그러나, SAM에서 제공하는 수직확대 기능과 지점간 시간/거리표기, 필터링기능은 현제 ASER에서 구현되지 않았다. ASER는 이벤트 레코더에 저장된 운행기록을

분석하는 툴로서, TIP의 실시간 모니터링 기능은 현재의 시스템 구조로는 구현할 수 없다.

## 5. 결 론

최근 안전한 철도차량 운행을 위해 철도차량의 속도와 운행정보의 기록 및 분석이 법제화 되고 있는 상황에서 운행속도와 제동정보 등 운행 중에 발생하는 운행정보를 기록하는 이벤트 레코더와 함께 이 데이터를 정확하고 빠르게 분석하는 소프트웨어의 중요성이 날로 커져가고 있다. 따라서 본 논문은 산업계의 요구에 따라 열차용 블랙박스인 이벤트 레코더의 데이터를 분석하는 소프트웨어를 설계하고 개발하였으며, 기록 당시 상황에 대한 빠르고 정확한 분석을 위해 운행정보를 그래프 뷰어를 비롯한 뷰어로 시간의 흐름에 따라 분석할 수 있도록 하였으며, 그 외에 열차 내에서 발생 되는 음성과 영상 데이터도 즉시 확인 할 수 있도록 개발했다.

개발된 ASER의 실험을 위해 실제 운행된 열차의 데이터를 이용하였으며 동기화된 뷰의 그래프와 그리드, 메시지, 16진 데이터, 오디오, 비디오, 시뮬레이션으로 그 결과를 확인하였다.

ASER를 이용하면 사고 발생 시 저장된 데이터를 토대로 보다 빠르고 정확하게 사고 원인을 규명하는데 도움을 줄 수 있으며 영상, 음성과 함께 당시 열차의 상태를 분석함으로써, 상황을 정확하게 파악하는데 결정적인 도움을 줄 수 있을 것으로 기대한다.

또한 평상시 운행기록을 분석함으로써 구간별 운행 패턴과 기관사의 습관 등을 파악할 수 있고, 이를 활용하여 교육함으로써 기관사의 조작 실수 가능성을 줄여 안전한 운행을 가능하게 하고, 비효율적인 운행 습관을 개선하거나 운행 효율을 높일 수 있을 것으로 기대한다.

그러나 정확성과 효율성을 더욱 증대하기 위해서는 SAM의 기능 중, 누락된 수직확대 기능, 지점간 시간/거리 표기 기능, 필터링 기능을 향후 추가해야 할 필요성이 있다.

## 참고문헌


[1] 최권희 외, "고속전철용 고장기록장치 시스템 설계에 관한 연구", 철도학회 추계학술대회논문집, pp. 29-33, 11월, 2005.

[2] 최권희 외, "중련편성 열차를 위한 효율적인 사건기록기 운영방안", 철도학회 추계학술대회논문집, pp. 1422-1426, 11월, 2007.

[3] 건설교통부, "교통안전시행계획", 5월, 2007.

[4] 이영훈 외, "차상신호 도입에 따른 신호기술 전망", 철도웹진, 2003년 9월, 10월 호.

[5] Railway Group, "Data Recorders on Trains - Design Requirements", Railway Group Standard GM/RT 2472, June 2002.

[6] http://www.eke.com/downloads/product_data_sheets/event_recorder/ - Product : Event Recorder

[7] Christopher J. Holliday, P.E. STV, Incorporated Philadelphia, PA, "A New Event Recorder Standard for Passenger Rail Equipment", IEEE, May 2005.

[8] IEEE, "IEEE Standard for Rail Transit Vehicle Event Recorders", IEEE Std 1482.1-1999, 26 June 1999.

[9] ALCATEL 외, "RTMS/ETCS - Class1 FFFIS Juridical Recorder-Downloading tool", Official Journal of the European Communities, 17 October 2005.

[10] 유병관, "고속철도차량 안전장치의 현황과 전망", 철도웹진, 2008년 1월, 2월 호.

[11] Faiveley, "SAM Installation and User Manual", September 2006.

[12] 송규연 외, "음성 통과 저장 기능을 제공하는 고속전철용 Event Recorder 연구", 철도학회 춘계학술대회논문집, pp. 1945-1950, 6월, 2008.



**장 동 욱(Dong-Wook Jang)** **[정회원]**

- 2005년 2월 : 호서대학교 컴퓨터공학과 (학사)
- 2007년 2월 : 호서대학교 컴퓨터공학과 (공학석사)
- 2007년 3월 ~ 현재 : 호서대학교 컴퓨터공학과 박사과정

<관심분야>
HCI, 시멘틱 웹, USN

**김 광 렬(Kwang-Ryul Kim)** **[정회원]**

- 2007년 2월 : 호서대학교 컴퓨터공학과 (학사)
- 2009년 2월 : 호서대학교 컴퓨터공학 (석사)
- 2009년 3월 ~ 현재 : (주)인지소프트 전자화문서 사업팀 근무

<관심분야>
USN, RFID, 임베디드 시스템

**손 석 원(Surgeon Sohn)** **[정회원]**

- 1985년 2월 : 인하대학교 전자공학과 공학사
- 1987년 2월: 인하대학교 대학원 정보공학전공(공학석사)
- 2007년 : 인하대학교 컴퓨터정보공학과 공학박사
- 1999년 9월 ~ 현재 : 호서대학교 뉴미디어학과 부교수

<관심분야>
인공지능, 무선통신망, e-Health

**한 광 록(Kwang-Rok Han)** **[정회원]**

- 1984년 2월 : 인하대학교 전자공학과 공학사
- 1986년 2월 : 인하대학교 대학원 정보공학전공(공학석사)
- 1989년 8월 : 인하대학교 대학원 정보공학전공(공학박사)
- 1991년 3월 ~ 현재 : 호서대학교 컴퓨터공학부 교수

<관심분야>
정보검색, HCI, e-Health, 시맨틱웹, 온톨로지