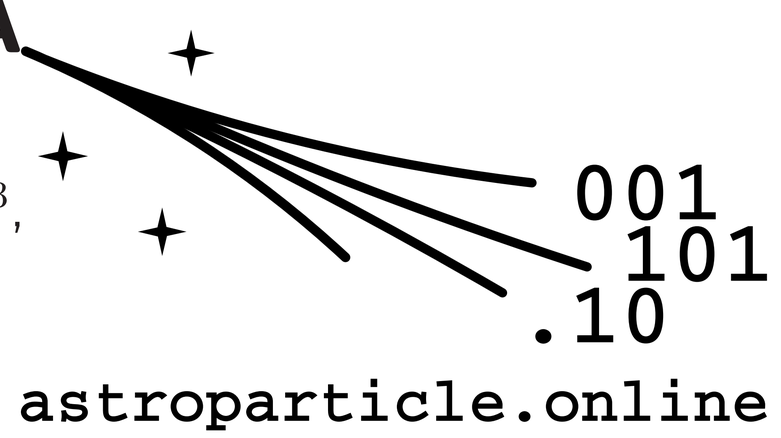
Specifying binary file formats for TAIGA data sharing and reuse

Elena Korosteleva[1,*], *Kryukov A.*[1], *Bychkov I.*[2,3], *Khmelnov A.*[2,3], *Mikhailov A.*[2], *Shigarov A.*[1,2,3]

1. Skobeltsyn Institute of Nuclear Physics, Lomonosov Moscow State University
2. Matrosov Institute for System Dynamics and Control Theory SB RAS
3. Irkutsk State University

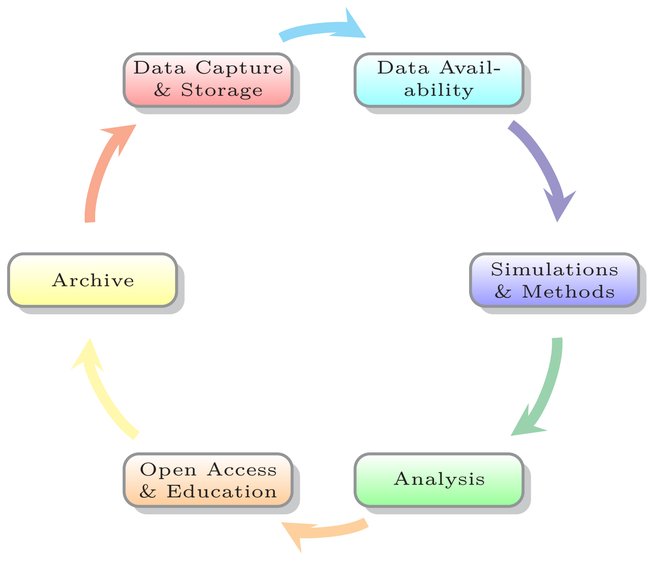
[*] elkrs@yandex.ru



1. Introduction

The on-going project, Karlsruhe-Russian astroparticle data life cycle initiative, aims developing an open science system to be able to collect, store, and analyze astroparticle physics data having the TAIGA (http://taiga-experiment.info) and KASCADE (https://web.ikp.kit.edu/KASCADE) experiments as the examples. One of the important issues is how to archive raw binary data to support their availability and reusing in the future. This work demonstrates ways for specifying binary file formats for TAIGA data sharing and reuse. There are several binary file formats used currently in TAIGA project. They provide a representation of raw data obtained from TAIGA facilities, including HiSCORE, IACT, T133, and Grande. The long-term preservation of raw binary data as originally generated is essential for rerunning analysis and reproducing research results in future. In this case, the raw data should be well documented and accompanied by some readers (i.e. software for parsing these data). The neglect of the issues may lead to the need for a reverse engineering of their file binary formats in future. The goal of the work is to provide human readable formal format description, as well as tools for serialization/deserialization and validation of the raw data. Some of the state-of-the art tools for formal describing binary data formats can provide a good enough solution for the issues of raw astroparticle physics data documenting and parsing. The work considers two of them, KAITAI STRUCT (http://kaitai.io) and FLEXT (http://hmelnov.icc.ru/FlexT). Both allow to express a formal specification of a binary data format in their domain-specific languages and to generate program code for parsing files of the format. The work presents our experience of using these tools for formal describing TAIGA binary data formats, documenting, and parsing library generation. This study can be interested in other experiments where raw binary data formats remain weakly documented or some parsing libraries for contemporary programming languages are required. This work has been financial supported by RSF (grant № 18-41-06003).

2. Motivation



Astroparticle Data Lifecycle

TAIGA formats:

- TAIGA-IACT
- TUNKA-HISCORE
- TUNKA-133
- TUNKA-REX
- TUNKA-GRANDE

Issue:

- No documentation, no open raw data access API
- How to verify and reuse raw data?

3. Solution

4. Kaitai Struct

Declarative: describe the very structure of the data, not how you read or write it

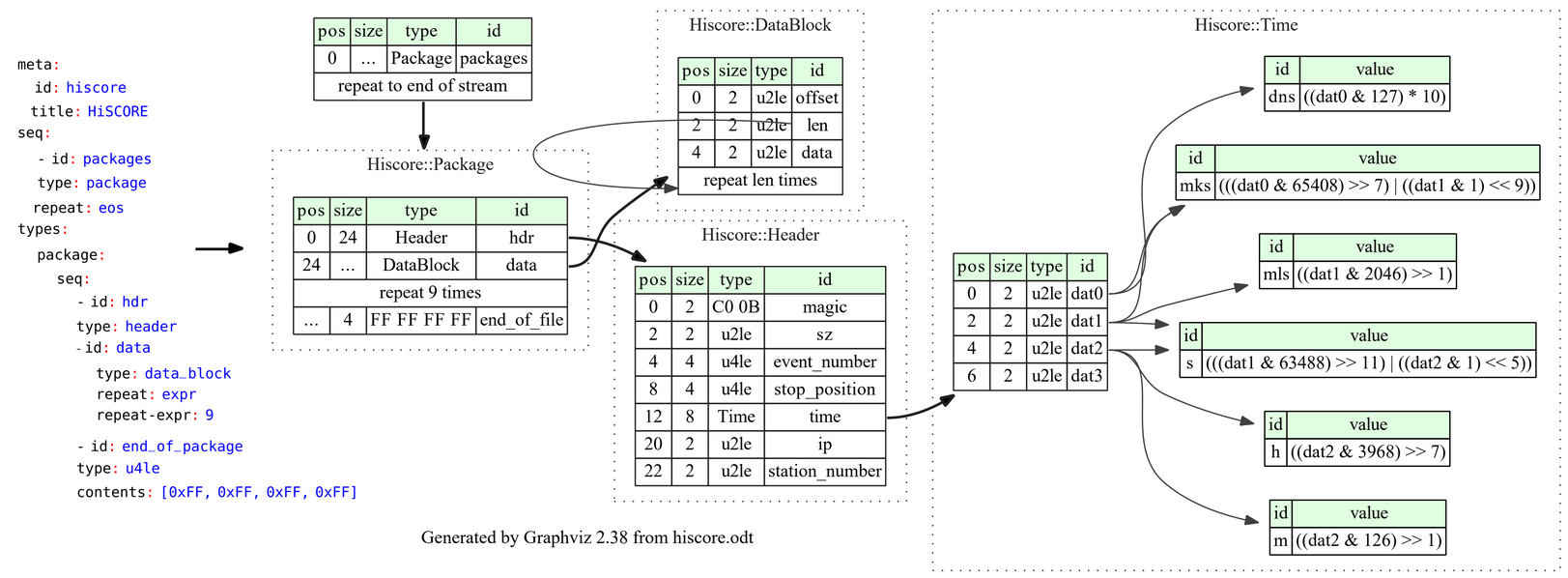
Packed with tools and samples: includes a *compiler*, an *IDE*, a *visualizer* and library of format specs

Free & open source: licensed under the following terms: GPLv3+ (Compiler and visualizer), MIT or Apache v2 (Runtime libraries)

Language-neutral: write once, use in all supported languages:

- C++/STL
- C#
- Go (*)
- Java
- JavaScript
- Lua
- Perl
- PHP
- Python
- Ruby

5. HiSCORE specification diagram



Generated by Graphviz 2.38 from hiscore.odt

6. How to use. C++ example

```
// Add KS runtime library
...
#include <kaitaistruct.h>
...
// Include generated class
#include "hiscore.h"
...
ifstream ifs(fileName, ifstream::binary);
// Make Kaitai Struct stream
kaitai::kstream ks(&ifs);
// Read HiSCORE file by generated library
hiscore_t hiscore = hiscore_t(&ks);
// Get all packages
vector<hiscore_t::package_t*>* packages = hiscore.packages();
vector<hiscore_t::package_t*>::iterator it = packages->begin();
// Print some infos
for (it; it != packages->end(); ++it) {
  hiscore_t::package_t* package = (hiscore_t::package_t*)*it;
  hiscore_t::header_t* header = package->hdr();
  printf("Event number: %d\n", header->event_number());
  printf("IP:           %d\n", header->ip());
  printf("Magic:        %d\n", header->magic());
}
```

7. Results

- TAIGA format specifications
- Reading libraries
- Examples for C++, Java, and Python
- The libraries tested on real data

- GRANDE, T133, and TREX tested on ≈ 89 000 files
- HiSCORE, IACT tested on ≈ 120 000 files

- 4 % of GRANDE, T133, and TREX files have BAD file format
- 0,6 % of HiSCORE and IACT files have BAD file format