Conceptual Modelling of Log Files: From a UML-based Design to JSON Files*

Evelina Rakhmetova[1], Carlo Combi[1], and Andrea Fruggi[2]

[1] University of Verona, Str. le Grazie, 15, 37134 Verona, Italy
evelina.rakhmetova@univr.it; carlo.combi@univr.it
[2] SIA s.r.l., Verona, Italy
andrea.fruggi@sia.eu

Abstract. In this paper, we describe an application of a recently proposed comprehensive UML-based (Unified Modeling Language) approach to the conceptual modelling of log files. On the real example, we built an ad hoc UML-based (class) diagram to represent the key features of the logs nested structure and generated an artifact (a template in JSON) based on ECS (Elastic Common Schema). We also describe plans for designing a specialized tool through a conjunction of the already developed artifacts. Presented work is a part of a broader study on the proposed initiative for the general concept of log files standardization. A clear structure of log data would allow more systematic development and more straightforward implementation and employment of the latest information systems, minimize anomalies, errors, and time delays.

Keywords: Conceptual Modelling · UML · JSON · Log Files · Elastic Common Schema · Modelling Tool.

1 Introduction

The stable work of information systems, with the constantly increasing complexity, and security of a tremendous amount of data, they are processing, profoundly rely on log files management. A log message is a piece of information produced during the work of the computer system or software, generated as a response to a running process or an action. The information pulled out of the log message provides an idea of the log message meaning and the reason for it being generated. Despite that modern log files management systems are powerful mechanisms for resolving issues of the IT industry generally, a growing number of custom solutions make every case rather particular [1].

Nowadays the practice of fast development and customization of applications leads to the situation when logs semantics is not always clear. Such messages do

* Our work is performed with the support and in the interests of the company SIA s.r.l., the provider of information technology solutions in the banking domain. The authors are particularly thankful to Daniele Spinelli (daniele.spinelli@sia.eu).

not give a distinct perception of processes and interfere with further analysis, hence, are not quite valuable. Even more challenging to keep track of different log file formats in large heterogeneous systems in which software and devices are dynamic in nature [2].

We declare an intention to limit heterogeneity in log files management by proposing a standardization of the log files through developing a conceptual modelling approach and a suitable tool. Conceptual data modelling provides analysts and designers with a high-level representation of the real world and an efficient way to communicate with each other. Such data models promote understanding of the real-world domain and enhance the ability to meet users' requirements [1, 3]. A key problem in log file design is the absence of a widely accepted conceptual model.

Based on the study and scientific literature review, we have determined a lack of studies on conceptual log files modelling [1, 2, 7]. There are standardized log files of systems and widely used commercial schemes, but there is no accepted methodology for the development of a log-based system, which could be applied everywhere [4].

In this paper we describe a comprehensive general approach to conceptual modelling of log data with a UML-like [5] graphical representation compatible with ELK stack (Elasticsearch, Logstash, Kibana) [6], elaborate on the first stages of work and then discuss the usage of the UML-based modelling approach and the developed python script (for generating logs templates and documentation). We also outline the future tasks for the tool development.

2 Applied Approach and Features to Conceptual Modelling Log Files

Our choice to establish log files conceptual modelling on extended UML-based (class) diagrams and on JSON is based on the following motivations:

- The UML graphical notation is commonly used over decades; it is structured and easily understandable by various users.
- Recently the JSON format has widely emerged as the most convenient standardized format for structuring data such as log files.
- JSON is relatively (with respect to other structured formats) compact, flexible, as almost every programming language can parse it, and human-readable.

2.1 Requirements for Logs

Since log files are mostly created automatically, as a minimum log data must include date/time stamp, description of the event and information unique to that event, in order to provide information to benefit further analysis, troubleshooting processes or data breach investigation. Information must be structured and suitable for running data analysis with the use of various tools.

2.2 JSON Log Files Formation

JSON logging provides more flexibility to the current logging system, especially when migration from the text logging format (as most common and unsettled format) to JSON can be simply performed. There is currently a vast number of frameworks and programming language drivers that support the translation of log data in JSON format if it was not initially the case. This shows the tendency of the industry to a standardized format for structuring such kinds of data.

One of the advantages - JSON is simple to implement in languages without built-in JSON functionality. It is important to highlight that on the metadata level ECS (Elastic Common Schema) [6] is specified through YAML format documentation (git repository), following, we will transform this for usability purposes in JSON format.

2.3 An Ad Hoc UML-based Diagram Modelling Approach

An application of the recently proposed approach [7] allows the representation of log files data structure in a more powerful way, thus providing a sound description of log-based systems. The extended UML-based modelling approach considers the use of suitable stereotypes to extend class diagrams [5] to represent the (mainly nested) structure of a log record.

A log record is composed of attributes and field sets, which group in their turn attributes related to the same feature the field set is representing. This gives extensive and clear to any user representation with a possibility to implement both top-down and bottom-up strategies on system development and/or adjustments.

Model Features The concepts of the class diagram model were taken as a basis [5]. Added features and extensions provide the support for an ad-hoc representation of log data. We explicitly highlight in the conceptual model that field sets and attributes partly coming from the ECS specification [6].

Composition associations are used to represent a proper nesting, where the nested parts may appear also within other parts, i.e., they are reusable. Field sets are represented through a class-like shape, where we distinguish three different sub-boxes, for core, extended and custom fields, respectively. An ad-hoc notation is also introduced for local nesting of field set. Other aspects considered in the conceptual data model for log files are types for attributes, associations between attributes, enumeration types, ECS metadata as categorization events, and an array of values.

For a demonstration, we show our extended diagram model obtained from the common log file record. It helps understanding complex data transformations. Fig. 1 shows a UML-based diagram for a single batch log of the custom application in the banking domain created with the implementation of the proposed conceptual model.

It is fair to highlight that the application is in use and currently acquiring log files are presented in text form, not structured accordingly, have numerous

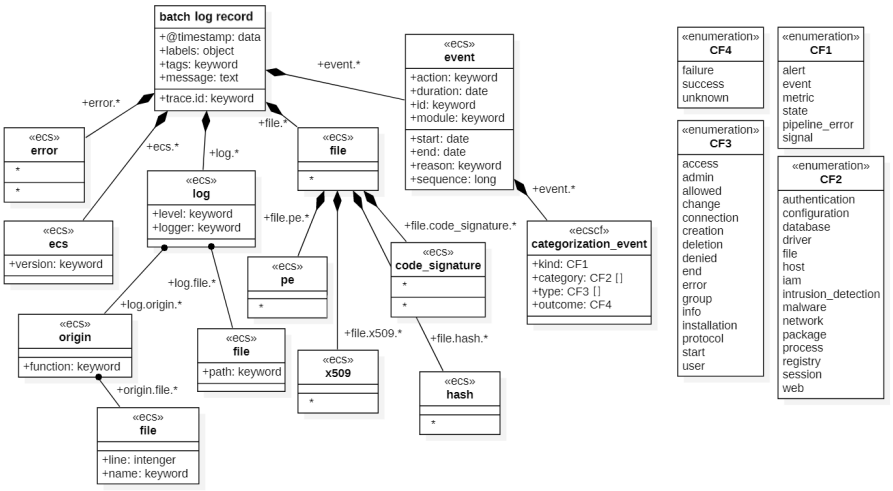

Fig. 1. Log file record of the bank batch application: graphical representation through the ad hoc UML-based conceptual modelling.

issues and completely unsuitable for proper monitoring and especially analysis. Applying our approach, together with an application owner, we succeeded to design a set of suitable log files records for the batch processes.

3 Towards the Implementation Process

As for the usability of the conceptual approach, we started by considering some real-world domains, from bank applications. Indeed, such kind of application covers various general event logging.

3.1 Python Script for ECS

The raw data were taken from the ECS repository opened for contribution at `https://github.com/elastic/ecs`. Originally ECS provides excessively many fields for log records, and only a few of them are needed to be populated for a certain case. Repository collects various files and tool templates, yet they do not provide universal applicability to any system.

We have chosen to maintain customizations by taking into consideration the tools provided by ECS and creating our own generator (python script, input and output files) to create relevant artefacts for the unique set of data sources. The script is running through the command line. Here are the main steps of the working process:

– As an input file, the current version of the ECS log fields set in YAML format is converted into JSON.

– Users can select the log fields from the set and include custom fields relevant to the project if it is needed.
– As an output, the artifact in the format of JSON file is obtained; it represents a sample template for a log record for the particular system.

Notwithstanding that the script is still in active improvement, it is already has been in use for several test cases of batch log files modelling. Fig. 2 provides an example of the case used as well for the UML-based diagram demonstration.

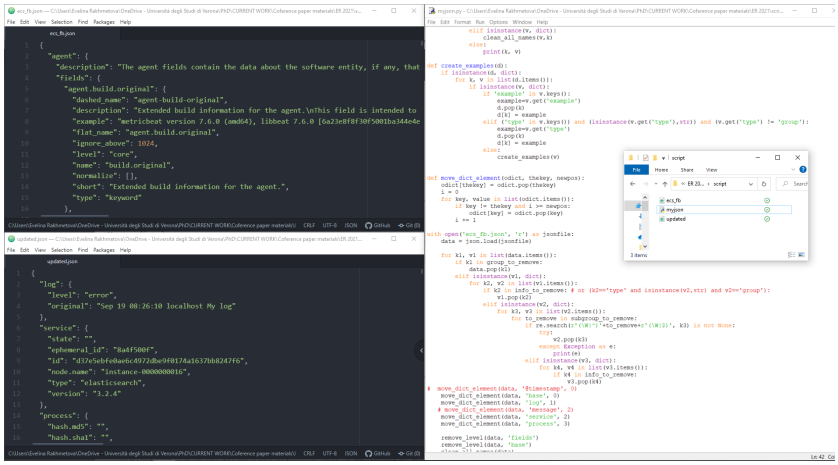

Fig. 2. The window with an artifacts generator code (right part), input file (the upper left corner; already formatted in JSON with all available fields in accordance with the current ECS version) and generated output file (the low left corner; the template for the custom log record in JSON format).

This script is a preliminary development for the future tool and has not been published in open source yet. It is one of the parts with the following that must relate to the UML-based graphical representation part.

3.2 Further Steps on Tool Prototype Development

At this point work not only propose the tool and step for its development but provides preliminary solutions. All together the organizational flow and conceptual model of possible architecture are showed on Fig. 3.

The tool is aimed at artifacts creation: log files structural templates (in JSON format according to defined fields from the YAML doc) and related documentation (which includes extended UML-based (class) diagrams).

The tool is aimed to provide for the conceptual modelling of log files from the beginning of system modelling or act as a supportive solution for redefining system logs. Yet in the second case, it is necessary to integrate loggers i.e. plugins

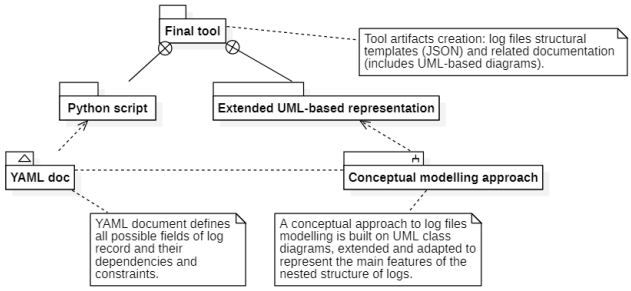

Fig. 3. An overall concept of the proposed tool architecture.

for the application (system) logging library to format logs into compatible JSON format.

4 Conclusion

As the result, we provided a comprehensive overview of our work on log-files modelling schema (in form of extended UML-based (class) diagrams) and developed preliminary instruments (python script) for log file modelling in JSON format (according to the predefined structure). In addition, we demonstrated our intention and actual steps for developing a comprehensive tool build based on the proposed conceptual log file models, which will provide both ad hoc UML-based diagrams as documentation and JSON formatted templates for log records.

References

1. Chuvakin, A., Schmidt, K. and Phillips, C. "Logging and Log Management: The Authoritative Guide to Understanding the Concepts Surrounding Logging and Log Management", 2012.
2. Nimbalkar, P., Mulwad, V., Puranik, N., Joshi, A. and Finin, T., "Semantic Interpretation of Structured Log Files," 2016 IEEE 17th International Conference on Information Reuse and Integration (IRI), 2016, pp. 549-555.
3. Combi, C., Oliboni, B., Pozzi, G., Sabaini, A. and Zimányi, E., "Enabling instant- and interval-based semantics in multidimensional data models: the T+MultiDim Model." Inf. Sci. 518, 2020, pp. 413-435.
4. Zhang, H., Lou, J.-G., Zhang, Y. and Chen, X., "Log clustering based problem identification for online service systems", 38th International Conference on Software Engineering Companion - ICSE'16, Austin, Texas, 2016, pp. 102–111.
5. OMG Unified Modelling Language (OMG UML), version 2.5.1, December 2017.
6. "Elastic Common Schema (ECS) Reference [master]", [Online], Available: https://www.elastic.co/guide/en/ecs/master/ecs-custom-fields-in-ecs.html, [Accessed: 10 June 2021].
7. Rakhmetova, E., Combi, C. and Fruggi, A., "A UML-based Approach to the Conceptual Modelling of Log Files", Technical Report Department of Computer Science University of Verona, 2021, in press.