C4R

Capacity for Rail

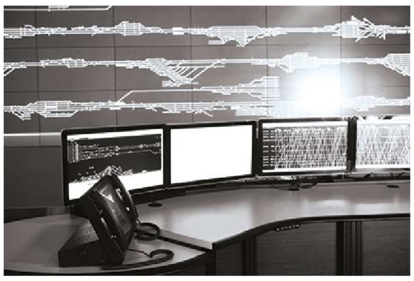
Towards an affordable, resilient, innovative and high-capacity European Railway System for 2030/2050

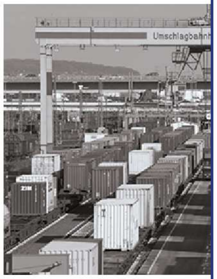
Recommendation for an Open-Source and Open-Interface for railway advanced monitoring applications

Submission date: 27/11/2017

Deliverable 44.3

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement n° 605650





Collaborative project SCP3-GA-2013-60560
Increased Capacity 4 Rail networks through enhanced infrastructure and optimised operations
FP7-SST-2013-RTD-1

Lead contractor for this deliverable:

- DB AG

Project coordinator

- UIC

Contributors

- Uob

1 Executive Summary

This Deliverable D4.4.3 presents a representative selection of Open-Source and Open-Interfaces which can be used as software components in a monitoring system.

Starting with a state of the art overview relevant aspects during the decision process when implementing a monitoring system are discussed in order to show the interdependencies and advantages/disadvantages of certain possibilities. Afterwards an exemplary list shows different libraries which can be used successfully in such a project. It’s impressive how great the variety of possibilities is. Even if they are non-commercial products, the Open-Source-Community offers products for professional needs. The speed of enhancement ensures fault free software which represents state of the art status –only aspects like graphics and report functionality are sometimes difficult to be implemented in a sufficient and efficient way.

It’s not the question, if there are software components available. It’s rather the question of choice than availability: “how does it fit together in my project?” Collection, assessment and distribution of monitoring data become cheaper and simpler. That ensures a wider use of monitoring systems in the railway environment.

2 Table of contents

3 Abbreviations and acronyms

Abbreviation / Acronym	Description
C4R	Capacity4Rail
km	Kilometre
EH	Energy Harvesting
EHM	Energy Harvesting Module
WPT	Wireless Power Transmission

4 Background

More and more advanced railway monitoring applications will be used to prevent critical damages by showing the evolution of the condition of the most important locations. Without this information it will be difficult to take right actions at the correct time.

For a successful implementation it is necessary to take several topics into consideration. Not only the type, the investment costs and the location of the monitoring system are of importance, but also the following processes of data transmission, data integration and data assessment. In this sense the terminology “monitoring system” includes both: monitoring applications as well as data applications.

Current monitoring systems are expensive and are often restraining an open data handling due to proprietary data protocols and data transmission. This limits a wider use in rail area.

Obstructions have to be low for a successful and wide use of monitoring systems. Many examples of other industries show how successful the Open-Source community generates software tools and libraries which are constantly enhanced. They reached a development status which enables a use in a professional environment.

To implement innovative monitoring technologies into existing networks, restrictions have to be low. Therefore these Open-Source and Open-Interface solutions have to be developed with the focus on:

- Easy to implement into existing monitoring systems
- Adaptable to user specific needs
- Extendible for future developments
- Using standard soft- and hardware components

5 Objectives

The general objective of this deliverable is to describe different aspects choosing an IT-environment for monitoring systems. The following main objectives are:

1. To describe the interactions between data acquisition, assessment, transmission, presentation and storage
2. To show different data protocol types and their specific advantages and disadvantages

This deliverable has been structured in sections. Section 6 describes the state of the art, Section 7 describes the processing of electronics, Section 8 the wireless data transmission, Section 9 the sensors used, Section 10 the installation zones identification of the two long-term monitoring systems for the different cases and Section 11 shows some results.

6 State of the art

Major steps took place in the last decade. It became easier and cheaper to collect and to exchange data. To be "on line" is no longer a matter of money, resources and special know how, but it becomes more and more an intrinsic functionality of technical products. Automatic generated reports, alarming if thresholds are exceeded, dashboards for condition monitoring with interactive and responsive graphic functions on a mobile device are some examples which show how naturally and simple it became to be part of the data driven world.

Interesting is, what kind of procedures and actions are running in the background ensuring this availability? The starting point is a collection of data – whatever content they're representing – and a procedure for transferring and distributing. There are different protocol types in use, e.g.

(1) binary files
(2) plain text protocols
(3) structured data set, like column orientated text protocols (e.g. comma separated values, csv), extensible markup language (xml), java script orientated data sets (json)

The last two ones are readable with any text editor (ascii), whilst the first one is only machine readable. All of them have their advantages and disadvantages and will be described in the following.

As they are not openly readable, binary files are often used in proprietary systems intended for use in isolation or where a supplier is interested in protecting their IP. Binary file storage and transmission is also comparably efficient due to the reduced overhead associated with headers and readability. The format was therefore historically used where storage capacity was limited, and is now more commonly associated with data transmission efficiency or with internal communications links within a larger single-supplier system. Additionally, binary files and plain text protocols are often used during a development phase. In most cases the emphasis lies on testing of equipment and not on structured data exchange. Normally some existing storage functions of measurement equipment or own developed software libraries are used to create result files. This leads often to non-distributable file formats (unless all parties are using the same software packages). Furthermore specific versions of software and firmware during the creation process limit the use of those files in future. It might happen, that old file versions are no longer readable by specific interpreter.

Hence: it is quite easy to create binary result files – but the danger of misinterpretation is quite high. The developer knows its properties and its construction, the data types and size of bits and bytes, the place and meaning of control characters. A very detailed specification is needed to interpreted binary values in a correct way, see IEEE 754-2008. Even different compiler versions may be taken into account.

This disadvantage is omitted with protocols which are readable in a standard text interpreter. Only the degree of complexity of data structure minimizes this benefit, but specialized text editor (e.g. xml interpreter) facilitate the handling of them.

Storage size as well as transfer bandwidth were limiting factors in the past. Therefore everybody was anxious to create most efficient applications – and the preference of binary protocols due to their limited storage size is understandable. This special "art" of programming was including a deep understanding of IT-hardware and network functionalities, but it's more and more diminishing nowadays. The limiting factors from the past are not as dominant as they were (besides really "big data"): CPU become quicker, storage is cheap, mobile and cable bandwidth increased enormously. A second topic is, that new software skills are needed e.g. for implementing graphical interfaces including webservices and programming new storage methods.

One example of a structured data exchange protocol used within the rail industry is railML. railML is a data exchange format developed by a consortium of railway companies, academic institutions and consultancy firms. The model is curated by the railML.org project team, and aims to continuously develop this format in order to facilitate its use in a wide range of railway applications. railML is published as a series of XML schemas holding subschemas, each of which encompasses a particular field of railway application:

- Common concepts and objects, sometimes not mentioned separately;
- Timetable (TT);
- Rolling stock (RS);
- Infrastructure (IS), both macroscopic and microscopic;
- Interlocking (from railML 3 on).

The railML family of models are well-known within the industry, and have served as the basis for activities in many EU-funded research projects, including significant activity in ON-TIME, and Capacity4Rail. For a detailed summary of railML and its use within the C4R project please see C4R project deliverables 3.4.1, and 3.4.2.

At time of writing, the IN2RAIL project (http://www.in2rail.eu) is drawing up recommendations for a canonical data model for rail, with railML 3 proposed as the first implemented serialisation.

7 Modelling phase

Every case has its specific demands. Therefore there is no clear answer to the question: "which protocol and which interface are the best"? The starting point is always challenging. In most cases not all aspects are known from the beginning, like "who are the partners? Who will use the data at the end in which way?" Some aspects are discussed in the following, which facilitate the user in their decision process finding the "best" data protocol. A sketch in Figure 1 illustrates some general components.

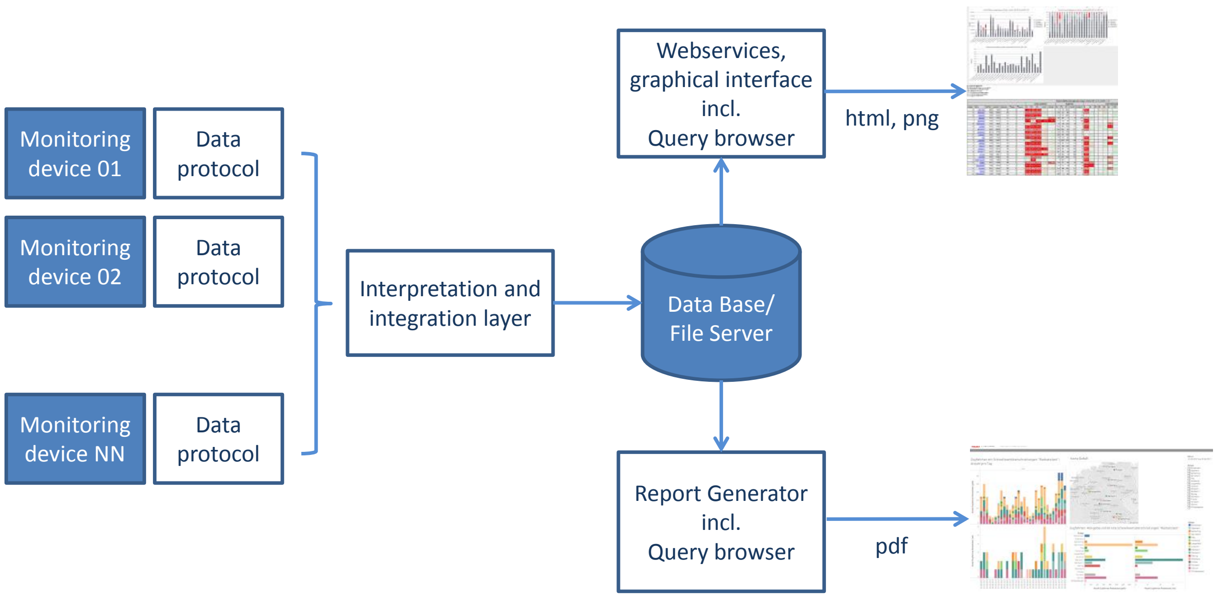


FIGURE 1: DIFFERENT SOFT- AND HARDWARE COMPONENTS FROM SENDER TO RECEIVER OF DATA PROTOCOLS

What is the aim?

It takes much effort to answer this classical and simple question. One is facing two aspects:

(a) superficial pragmatism

(b) elaborate details.

One has to find a balance between knowledge as well as juvenile unconcern, because future possibilities and needs might be unpredictable from today's point of knowledge.

Good practice is to investigate all steps from "who is the receiver?", "what kind of answers are looked for?" to data acquisition, assessment criterias and their representing values, preferred data transmission, soft- and hardware preferences, until the process of presenting the result and data storage. All aspects are then collected and assessed in a specification sheet in order to document the decisions and their background.

What kind of soft- and hardware environment shall be used?

Often there is a difference between operating systems of data acquisition system (sender) and those one on the client system (receiver), e.g. linux and windows. Some small differences concerning control characters in text files have to be taken into account, because they are defined differently (e.g. control character for line feed). If a collaboration between different software developer is planned, those small differences have to be clear. Some linux-commands which might be used in php-scripts are not available on a windows machine.

Is it possible to define the data model by one's own or is it specified by an external supplier?

As written earlier, a specification sheet is needed in all cases. This paper is a basis for all next steps. It depends on the business relationship with external supplier, if it's wise and efficient to overtake this responsibility to an external partner. In some cases of very strong partners it might be very difficult and costly to implement any changes in the data structure after a revision process. Especially when one self is hosting the data bases it would be recommended to define the data model also by one self. It is often seen, that data base structure and structure of transmitted data are closely related with each other.

Are there any standard approaches to data models?

Ontologies are data models that formally describe a particular problem domain or world view in a machine-interpretable way. By creating ontologies and associating real-world data with them, it becomes possible for computer systems to infer new knowledge in the same way that humans might use common sense when considering a set of facts. This inference aids data integration by making data implicit within a model explicit, and simplifies information systems by reducing the amount of logic required in individual applications when interpreting data.

The Rail Core Ontology (RaCoOn) is an ontology model specifically tailored for use within the railway industry. Although initially developed with the representation of signaling and rail infrastructure in mind, the model rapidly developed into a general model for the railways, including a core of generic railway concepts with extensions capturing particular subdomains (infrastructure, timetabling, rolling stock etc.) and an upper level model to define concepts used more broadly than rail (e.g., transport). By drawing on other publically available ontology models, for example the Semantic Sensor Network, RaCoOn can be used as the railway-specific based layer for describing the location of more general devices, such as sensor nodes attached to the infrastructure. For further details on the RaCoOn model, see C4R deliverable 3.4.1, while details on appropriate data architectures for use with ontologies, including use cases involving streaming asset data, can be found in deliverable 3.4.2.

How big is the probability for changing the data content in the future?

Normally there are two settings: Either an existing application shall be modernized by using recent technologies or one starts from scratch. The former case is well established including a description of involved data sets, their graphic representation, user handling and interactivity as well as storage needs. One major key is the data representation: what kind of data is needed? This topic is fixed due to the long term experience, while it might change during the introduction phase for the latter case.

And here one has to find an optimum. Some data protocols allow an integration of new data fields quite easily, e.g. xml and json. Even a mixture of "old" and "new" datasets can be used in parallel. Only the data interpretation algorithm has to be clever enough, to deal with missing data fields (which is normally not a big deal). Column orientated formats (like csv) doesn't allow this flexibility so easily, because there is some more work needed when programming the data interpretation algorithm. The worst cases are binary protocols with fixed header structures integrated by external software. The data integration process is often optimized which makes it in most cases very difficult to change anything.

Does anybody should be able to read the dataset quite easily, e.g. with a normal text orientated editor for checking the content or for information purposes?

It depends on the complexity of data structures and the predicted volume of data. The more complex the data structures are the more difficult is it to check it manually. Even if a special editor may support a structured view (e.g. xml-editor), nested data are difficult to assess on a screen. Often it's better to create a combination of graphical and tabular view on interested data sets, so one may check if desired data are at the correct place in the data set. Only if errors in data interpretation occur it might be helpful to look into the raw-data.

What are the roles and their needs of sender and receiver?

All definitions concerning data protocols and interfaces should begin with the needs of the end user (= receiver): "What kind of answer shall the product be able to provide? What kind of basic functionalities are needed, e.g. refresh period of transferred data, user friendliness for maintenance of monitoring systems (e.g. what kind of data are needed for representing a status of the monitoring device?), does it has to fit in an existing software environment?," etc. At the end it may happen that some wishes or needs are not realisable by the provider of the monitoring devices, but it's better to know the limitations from the beginning of a development.

A collection of end user's needs is a relevant basis for the next discussion about technical questions like: "how are the needs implemented?" What kind of technical solutions can be choosen (or financed?)". Sending may happen at two layer:

(a) Sending of data protocols from monitoring devices

(b) Sending of results from data bases/file server

Two different parties may be responsible for those two tasks. Their view differs a lot. The first one is closely related to hardware components of the measurement system. They have to integrate all different measurement values from different devices provided on different interfaces into one software and one resulting data protocol.

The second one has to collect information from different data sources stored in different data bases/files, calculate and/or aggregate the needed representing result values into a presentation format (e.g. html, png, pdf) and transfer it to the end user. Here a close discussion with the end user about needed values, aggregation levels, graphs, tables, texts, layout of the presented format is relevant. What kind of interactivity should graphics offer to the user on websites? Today some graphic libraries offer a huge variety of possibilities to show additional information when hovering with the mouse over the graphical items – or even enable hyperlinks to new dashboards.

What happens with a data set? Is it just stored in a file system, shown on a web front end or is it even integrated into a database?

As described earlier, the beginning phase of a project is most relevant. All needs, wishes and ideas have to be integrated in a specification sheet. The question of storage has a major influence on the subsequent data processing.

Different scenarios are possible, like:

(a) Datasets show only a state, but are not needed for answering long term questions

(b) The number of datasets is small and no in-depth analyse is necessary

(c) There are a great number of different values and protocols on a long term basis

The first scenario is typical for small applications with little data amount. Here it's often more simple to convert values of the data protocol directly to graphical representations and to store them afterwards.

The second one is similar to it, but the difference is to store the data protocols on a small PC. There are some tools coming from big data analysis which are able to work with files containing relevant information and no database is needed. But this works only, if the number of interested values is small and the data structure is simple.

A standard procedure is described in the last scenario. A lot of information has to be dealt with, including a long term perspective. The definition of the database itself is closely related to the answers. The better this fits together, the quicker the queries will run. Sometimes it makes sense, to calculate regularly some aggregated results and to store in a new data base table, instead of calculating it by a query of the customer, even if it's not the academic way of not storing data twice.

What about data compression?

Even if size or costs for storage are no longer a limiting factor it is wise to think about this topic. Not all locations will have a good mobile connection for transferring the data protocol to a central receiving service. Therefore packing of them before sending is recommended (e.g. zip). If data protocols are stored as a backup the amount of files grows with time and needs a procedure for storage handling. A backup of central file server or data base is needed as well. What kind of period should be stored for analysis in future? It is recommended to organise some data base tables for a certain period, e.g. per year, quarter or month, depending how much data is stored in this period. Later on one can extract those ones which are too old and put them into a backup system. If needed, one can integrate them later again. This will help to keep the performance on a high level.

What about encryption?

Even if the safety level of the monitoring system might be little an undisturbed and safe data transmission is needed. Information between sender and receiver shell be only known to both of them and not to others who are not involved. Therefore encryption is a topic – both data and communication. Due to the quick development in this field it’s not easy and wise to give here irrevocable instructions. At the end, there is no guarantee for a perfection situation. Depending on the effort, there will be one or the other possibility for intrusion. But some basic rules shall be mentioned here:

1. Every sender is using fixed IP-adresses together with its own identification (username, password), so a central receiving service can follow who is communicating at which time.
2. Firewalls can be configured easily by knowing all partners and their properties (e.g. source and destination IP-address, kind of application, their network protocol and port). Partners might be persons as well as machines. It’s obvious that fixed IP addresses facilitate the firewall configuration
3. Usernames and passwords are changed regularly
4. If possible, sender and recipient take part in a closed network environment (APN). There are special subscriber identity module cards (SIM) available who are part of an own APN. Although the communication runs over the public internet, only participants of this APN can follow the communication.
5. All transmitted data might be encrypted.

What kind of encoding is used?

A character is a minimal unit of text. A collection of characters used in a certain language is called a character set. There are app. more than one million different characters used in all worldwide languages. ASCII is a very popular charset with 7 bit length, where every character and some symbols are represented by an integer value.

For a correct understanding between different software programs the encoding has to be transmitted as well. That mean, that the data content, the encoding on a web-site and the data base should use the same encoding. Otherwise some misinterpretation might occur (e.g. german Umlaute äÄöÖ, Nordic vocals øœ, etc.). Today in many cases a UTF-8 charset is used.

What about compatibility with existing systems?

Many organisations operating within the rail industry already have an extensive collection of monitoring and other IT systems. The addition of further, disconnected, solutions are often therefore seen as a complication or hindrance, rather than a source of benefit. Increasingly, these organisations are looking to break down the barriers between these disparate monitoring (and other IT based) systems such that a more connected approach can be taken. In some cases this involves the internal identification and exchange of existing data, while in other cases it involves ensuring that systems can make their internal data available on a technical level. It is therefore important that new sources of data take into account the existing data storage, exchange, and representation formats. It is also good practice that new systems are designed to inherently be compatible with existing IT infrastructure, or, where 3rd party outputs are required from the supplier side, that the solutions have additional data exchange capability that allows the systems to be incorporated into existing infrastructure.

8 Proposal of Open-Source and Open-Interface

A monitoring systems (see Figure 1) consists of different components like

(a) a monitoring device for data acquisition

(b) a transfer layer for communication in order to send/receive the data protocol

(c) a receive and integration service for data protocol

(d) a data storage system

(e) a query browser and result presenter system

Open source software components are available for some of them. They will be described in the following.

A monitoring device is a mixture of different hardware and software components. The interaction of them is often very complex. Professional measurement software tools are used in most cases to combine data from different devices and interfaces. Sometimes some provider specific software tools are used to generate the resulting data protocol and to communicate with the receive service for transmitting the data. Therefore open source tools are used very seldom.

More tools are available for all other components (b) … (e). The following table give an overview of currently often used software. This list is not complete and gives only recommendations

Table 1: List with examples of Open-Source Packages for different purposes

Role	**Example**	**Comment**
Encrypted communication with remote access	TightVNC viewer	see https://sourceforge.net/projects/vnc-tight/
Encrypted communication	OpenSSL	see https://www.openssl.org/
Script language for creating web-pages, sql queries and graphics	php	see http://php.net/
JavaScript utilities for own websites (e.g. sorted tables, calendar, tabs, menu, etc.)	jquery	see https://jquery.com/
Script languages to create receive services, automatic routine for database query generation and sending reports	StrawberryPerl for windows	see http://strawberryperl.com/
Another script language with a lot of mathematical and graphical libraries	python	see https://www.python.org/

Basic graphic libraries	gd gnuplot	see https://libgd.github.io/ see http://gnuplot.info/
Java	java	see https://www.java.com/en/
Java Graphic libraries	jpgraph d3.js rickshaw	see http://jpgraph.net/ see https://d3js.org/ see http://code.shutterstock.com/rickshaw/
Statistical computing (statistical functions, graphic libraries, sql queries)	R	see https://www.r-project.org/
Dashboards based on R-functionality with interactive graphics	Shiny, Plotly	R-Studio is needed, see https://www.rstudio.com/products/shiny/
Big Data Analysis toolbox	Knime	see https://www.knime.com/
Package with webserver, DataBase, php, perl, etc.	xampp	see https://www.apachefriends.org/de/index.html
DataBase	MySQL MariaDB	see https://www.mysql.com/de/ see https://mariadb.org/
Report generator	BIRT	see http://www.eclipse.org/birt/
Java Script Object Notation	json	see http://json.org/
Toolbox for creating and reading xml-files	php simple xml perl simple xml	http://php.net/manual/de/book.simplexml.php see https://metacpan.org/pod/XML::Simple

Open source libraries are available for different types of data protocols, like xml or json. xml is widely used, there are libraries available for almost every programming language. json is closely connected with java and ensures an easy communication e.g. for web frontends. xml is more strict and generates more overhead compared to json, but there are testroutines available which check the integrity of a xml data protocol, if it's content fulfil the specification.

Open Source software libraries are frequently used for the receiving and integration service in connection with databases and websites. A common software package is xampp which offers a webserver, a data base, php as a programming tool for creating webpages, queries and graphs as well as perl. perl is a script language used for managing periodic tasks like data integration within a certain time interval, creating queries which run every night, etc.

One big issue are graphic tools. It's quite easy to generate sql queries and tables with their results (e.g. using jquery). More effort lies in a graphical representation. Every graphic library demands a certain data format and has its limitation in the complexity (but even a correct placement and a certain type style of a legend might be difficult or even impossible). Standard business orientated graphs like bar plots, line plots are often possible, but even a secondary y-axis or a free mixture of bar and line-plots might be impossible with some graphic library. Sometimes it takes long time to test examples given in some internet forums, to read insufficient documentations – and then one notices, that this graphic style is not creatable. Even more effort is needed, if interactive graphs want to be used. Nowadays some javascript based graphic libraries (e.g. Rickshaw) offer the possibility to show additional information when the user hover with the mouse over some graphic entities. Here the programmer has to ensure this additional data in form of json datasets. The programm code starts getting more complex if in this case a forwarding with hyperlinks is wanted.

9 Conclusions

A lot of different Open-Source libraries and interfaces are available and can be integrated in a project of a monitoring system. The user can choose among a great variety depending on the overall concept and some preferences. Some of the products are well known and reached a status as standard software, like R, MySQL, php + html, perl, JavaScript, etc.

The Open-Source-Community is very active. Even if non-commercial software packages are offered, sufficient user friendliness is gained in most cases: There are documentations, “how to”-tutorials, videos with explanations, user forums with question and answers, blogs with certain examples available. Most questions are treated in the one or the other way. Complete scripts with typical code can be downloaded and easily adapted to one’s own project. It’s interesting to see, how quick new releases are offered and how stable those libraries are running.

There are Open-Source solutions for nearly all components of a monitoring system available. Besides the monitoring device itself all other tasks can be overtaken and guaranteed by them. Some of them are working quite easy and there are very frequently used products available, like protocol generation, data transferring, data integration into a database, query generation, statistical analysis and website programming. A lot of stable product can be chosen. More effort is needed for creating graphs and reports. The more complex a graph should be, the more coding is needed. Especially scientific orientated graphs are challenging. The same might be said for generating reports. There are some Open-Source projects, but in most cases they generate business orientated reports. Complex number crunching is not their strong point. In some cases, professional tools (e.g. Matlab, Tableau) offer here a better performance.

It is obvious, that the development of those software libraries is not predictable. Today’s standard might be the wrong horse in some years. The speed of development, especially for webservices, is enormous. New protocol types, data handling and communication processes are developed quicker and quicker. Certain flexibility when implementing a monitoring system is needed, so that a future change of some software components is done easily.