PSFC/JA-17-42

MDSplus Yesterday, Today and Tomorrow

T. Fredian[1], J. Stillerman[1], G. Manduchi[2], Rigoni[2],
K. Erickson[3], T. Schröder[4]

[1] Massachusetts Institute of Technology, 175 Albany Street, Cambridge, MA 02139, USA
[2] Consorzio RFX, Euratom-ENEA Association, Corso Stati Uniti 4, Padova 35127, Italy
[3] Princeton Plasma Physics Laboratory, Princeton, NJ 08543, USA
[4] Max-Planck-Institut für Plasmaphysik, D-17491 Greifswald,, Germany

October, 2017

Plasma Science and Fusion Center
Massachusetts Institute of Technology
Cambridge MA 02139 USA

This work was funded by DOE Grant Number: DE-SC0012470.

MDSplus Yesterday, Today and Tomorrow

T.Fredian[1], J. Stillerman[1], G. Manduchi[2],
A. Rigoni[2], K. Erickson[3], T. Schröder[4]

1 Massachusetts Institute of Technology, 175 Albany Street, Cambridge, MA 02139, USA
2 Consorzio RFX, Euratom-ENEA Association, Corso Stati Uniti 4, Padova 35127, Italy
3 Princeton Plasma Physics Laboratory, Princeton, NJ 08543, USA
4 Max-Planck-Institut für Plasmaphysik, D-17491 Greifswald,, Germany

MDSplus is a data acquisition and analysis system used worldwide predominantly in the fusion research community. Development began 31 years ago by a collaboration of software developers who were charged with providing a data acquisition system for three new fusion experiments under construction: CMOD at MIT, ZTH at LANL and RFX at Padova, Italy. The design of MDSplus combined the functionality of MDS (MIT/Model Data System developed at MIT for the Alcator and Tara fusion experiments) with new features suggested by the developers from the other laboratories. The development of MDSplus used a RAD (rapid application development) approach before RAD became a mainstream methodology. MDSplus was implemented and ready for the initial operation of CMOD in 1991. Since that time, many other fusion facilities started using MDSplus for data acquisition and/or for exporting their data to other sites. Today MDSplus is still used around the world for fusion energy research, space exploration and other fields of science and technology. Work on MDSplus continues to enhance its capabilities, support more platforms, and improve its reliability. It is anticipated that MDSplus will continue to provide valuable tools for the fusion energy research community. This paper describes some of the history of the MDSplus software, the work that is currently underway, and the plans to enable MDSplus to continue to be available and supported long into the future.

Keywords: Data Acquisition Systems, Data Management, Data Formats, MDSplus

1 Introduction

MDSplus[1][2] is a Framework for data acquisition and management of scientific data. It consists in a collection of libraries and applications used for acquisition, access and storage of scientific data. MDSplus functionality can be summarized as follows:

- ***Data organization and storage***: the MDSplus data access layer defines a variety of data types for scientific applications. Data types such as scalars and arrays are enriched with derived types such as signals, to represent both samples and timestamps. Metadata can be defined, e.g. to associate a description or a validation procedure to a given data item. Hierarchical collections of data items can be stored in pulse files, possibly distributed across multiple computers. Pulse files are created by cloning a template model, called experiment model, that defines the data structure

and contains as subset of data items, that is, the description of the experiment configuration and all the information about the experiment that is known in advance. The cloned pulse file will be enriched with experimental data collected during the experiment sequence. Other data systems, such as HDF5, provide support for hierarchical data organization for a variety of data types, but MDSplus provides two unique features: the use of generic expressions data representation and Multi Reader/Multi Writer (MRMW) ability. Unlike the other data system where a data item is an instance of a given data type, in MDSplus every data item is an expression, that is, a program that returns a given data type. Expressions can be as simple as the definition of a scalar value, or be represented by hundreds of lines of code specifying how the returned data item is built, based on other information, normally stored in the pulse file itself. Internally. Expressions are stored in pulse files as the parse tree of the corresponding program (in case of a simple data, the parse tree becomes the data item itself), and every time a data item is read in MDSplus, the corresponding expression is evaluated on the fly. Expressions add an expressive power in data representation that cannot be provided by other data systems and they are widely used to describe the whole acquisition chain for acquired signals and provide a way to dramatically reduce the size of pulse files by eliminating redundant data. For example, a set of signals acquired using the same timebase can refer to a single timebase item in their expression definition instead of storing the whole time array for every signal.
MRMW ability is another unique feature of MDSplus. Data systems supporting structured data typically preserve integrity in data access by limiting file access to a single writer, and often even multiple readers are not allowed. This represents a serious limitation in large data acquisition systems where different actors store data concurrently in the pulse file. MRMW is provided in MDSplus using a locking mechanism that is implemented in the lowest data access layers and is therefore available to all the MDSplus components.

- ***Access to the data and metadata from a wide variety of programming languages and utilities***. MDSplus provides Application Programming Interfaces (APIs) in a variety of programming languages including C++, Java, Python and Matlab. The same data semantics are defined in all APIs, expressed as a set of classes providing methods that allows all the expressive power of MDSplus to be exposed to users. Users can access data in a straight way, just calling the data() method that evaluates the corresponding expression on the fly, getting the data item they are interested in. More experienced users can develop their own expressions to efficiently organize data semantics.

- ***Remote data access using a variety of transport mechanisms***. This is another unique feature of MDSplus, that is, the possibility of acting on remote data preserving the same interface as for local data. In this way the implementation of remote data acquisition systems is greatly simplified. The developer does not need to worry about data transport. This is achieved by implementing in the lowest data access layers a transport mechanism based on a specific high level protocol, called mdsip, initially built over TCP/IP and currently available in a variety of transmission protocol.

Remote data access using mdsip turns out to be much more efficient than using other protocols for distributed files such as NFS. The reason is that, unlike general purpose distributed file systems, network traffic in mdsip is optimized and targeted to the specific MDSplus requirements in remote data access. This configuration, called *Distributed Client*, is enabled by setting a few environment variables that specify whether data access is local or remote and where data are stored. Another remote data access configuration supported in MDSplus is called *Thin Client*. Applications send expressions to a server for I/O and evaluation and results are sent back to the client. This optimizes the number of network transactions for data access and it is best suited for remote data access in Wide Area Networks where latency in transaction becomes an issue. Thanks its flexible and efficient remote data access layer, MDSplus is now the standard de facto for remote data access in fusion experiments.

- ***Data-driven workflow engine for data acquisition and automated analysis***. Unlike traditional data systems that store configuration data and acquired signals in separate databases, in MDSplus the same pulse file stores all the experiment-related information, not only including configuration and experimental data, but also the description of the actions carried out during the experiment sequence and of the associated scheduling information. The Action data type specifies *what* should be executed (a program, a procedure, and I/O routine, etc), *when* it should be executed (i.e. its relationship with other actions in the sequence) and *which* actor should execute this action.
- ***Integration of user-provided components***. MDSplus provides the *device* pattern that is a set of classes and rules that allow users integrate specific data acquisition components. Such components, normally interacting with specific hardware, can be integrated in the framework and becomes an integral part of MDSplus. For example, they are recognized by the workflow engine and the corresponding actions scheduled during the experimental sequence.

This paper will discuss the history leading to the development of the MDSplus data system, the current work underway to improve the quality and functionality of the software and the prospects of MDSplus continuing to be a useful tool for fusion energy research in the future.

2 The History of MDSplus

2.1 First there was MDS

In 1982, two MDSplus authors, Tom Fredian and Josh Stillerman, joined the Alcator C fusion experiment team at the Plasma Science and Fusion Center (PSFC) at the Massachusetts Institute of Technology. They were tasked with providing computing assistance for engineers and scientists of the two experiment groups.

The techniques for acquiring and storing data performed by scientists and engineers at the Alcator C experiment were very primitive compared with today's standards. The main source of experimental data was through digitization of 128 signals recorded on a large analog storage drum using a single CAMAC digitizer to digitize four signals at a time and write the data to a disk file. In addition to the analog drum a few other CAMAC digitizers had been used to record measurements. For this data the scientist responsible for the measurements developed or copied and modified a specialized program that would read the digitizer and store the data into a disk file in a format only known by the scientist who wrote the program. The other commonly used mechanism for storing measurements was the use of a Polaroid instant camera to take photos of oscilloscope displays which were then often digitized by hand by scientists.

Based on the previous experience with process control system software they started developing a data system where the data could be stored in a central database and the scientists and engineers could use a common set of tools to record and access experimental data. This new system was later given the name MDS[3] which originally stood for Model Data System but was later called MIT Data System.

The MDS data system was developed entirely using the FORTRAN programming language to be used on the Digital Equipment VMS operating system. Digitized data from CAMAC modules along with timestamp information were stored in a datafile and could easily be accessed using a unique signal name given to each measurement. Software modules written to read a particular type of CAMAC digitizer were developed to enable a scientist to configure data acquisition with the digitizer simply by specifying a few device specific settings, the signal names to use for the channels being recorded and some timing information to construct a timebase for the measurements. In addition, the system provided a simple mechanism to do a linear conversion with a coefficient and offset to convert digitizer counts into measured voltages. With MDS, all of the scientists and engineers could access all of the measurements recorded after each pulse of the experiment using the same software and interactive tools for visualizing the recorded signals. Some other features of MDS are worth mentioning. The data access code of MDS supported concurrent multiple data writers and readers. This allowed new data to be stored by multiple programs simultaneously at the same time as previously stored data was being accessed by users. MDS also provided a lossless compression of the data greatly reducing the disk storage requirements. The system relied on dynamic image activation and call by name to provide extensibility, avoiding the need to recompile / relink the distributed code.

The MDS system developed in less than two years turned into a very useful tool for the scientists and engineers at MIT and soon many other fusion research facilities around the world learned of MDS and began using it on their experiments as well. By 1987 MDS was in use at PPPL, ORNL, UCLA, SNL, U of Washington, Columbia University and U of Wisconsin (United States), ANU (Australia), NIFS (Japan), KTH (Sweden), and NFRI (South Korea).

Much of the acceptance and success of MDS was that it was a collection of tools that could be used by any fusion experiment facility requiring only site specific configuration setting modifications without modifying the MDS software.

2.2 Then there was MDSplus

In 1987, three new experiments were under construction: Alcator C-Mod at MIT, RFX in Padova, Italy and ZTH at Los Alamos National Laboratory in New Mexico. The scientists at RFX had learned about MDS and thought they might want to use it on the RFX experiment but they had some ideas for additional functionality that they wanted in their data system. The computer staff at RFX proposed a collaboration between the developers at MIT, RFX and ZTH to develop a new data acquisition system for the new experiments building upon the existing features of MDS.
At the end of 1987 a development project was initiated to design and implement this new system for the Alcator C-Mod, RFX and ZTH experiments which would provide much more capabilities than the MDS system. More developers started working on the new system that was given the name MDSplus.

While MDSplus retained some of the original features of MDS it was a completely new design and written almost entirely in the C programming language. Like MDS, it provided concurrent reading and writing to the data store by multiple processes. It also introduced many new features including the ability to store many more types of data such as configuration settings, comments, measurement data, analysis results and other metadata organized in a hierarchical tree structure so all data related to an experiment cycle could be easily navigated and documented so the information would retain its meaningfulness for years after it was recorded. In fact, each node in this tree structure contains automatically recorded metadata such as who updated the contents and when, and the size and datatype of the contents. The tree structure can also consist of multiple separate trees which can be dynamically linked together with each tree having its own set of file access permissions. New diagnostics or analysis systems can be represented in separate tree structures which can be linked into one main tree structure for the entire experiment. The new system also provided a powerful built in expression evaluation feature. Using this expression language, new signals can be defined as complex mathematical expressions referencing other nodes in the MDSplus tree which would be evaluated when accessed by the scientist. When MDSplus was first designed, object oriented programming was just in its infancy. The first international standard for the C++ programming language was not published until 1998. While MDSplus was implemented in C, a lot of object oriented features were included in the design of MDSplus. In particular, every data item managed by MDSplus is internally represented via a descriptor making it self descriptive. Consequently no static type definition was required to handle data, and every data item instance was able to properly handle the set of supported functions. For example, every data item was able to handle the data() function: a scalar type simply returned its value, a more complex expression could activate a sequence of steps to carry out the represented computation and eventually return a result. Thanks to this organization, the integration of Object Oriented languages and interfaces carried out in 2008 turned out to be almost straightforward.

The design and implementation of both MDS and MDSplus used a RAD (Rapid Application Development) approach[4][5]. There were no formal requirements analysis or structured design performed. Instead requirements were gathered from the user community and discussed by the developers and prototype implementations of modules were developed and tested by the developers and the users. This approach is particularly applicable to a research environment where the needs of the scientists and engineers continue to evolve during the lifetime of the experiments they are working on.

The new MDSplus data system was ready for initial use by the end 1991 where it was first used to store data from the first operation of the Alcator C-Mod experiment. Shortly after that time, the RFX experiment in Italy also started operation using MDSplus. The ZTH experiment project was terminated before completion. It should be noted that after nearly 26 years have passed the original data stored during the startup of these two experiments is still accessible using current MDSplus software releases

In 1995, a scientist from the University of Maryland was developing a diagnostic for use on the Alcator C experiment and was interested in remotely accessing data stored in MDSplus from his Digital Unix computer system. The MDSplus core developers quickly developed a remote access protocol for accessing MDSplus data based on the TCP/IP network protocol that just became widely used in the early 1990's. This feature of MDSplus, called MDSIP, turned into one of its most valuable assets enabling scientist and engineers to remotely access MDSplus data anywhere on the Internet using a very simple API. Eventually this built-in capability made it easy to demonstrate remote control of the experiment as early as 1997[6].

Since the initial design and development of MDSplus the operation of fusion devices have continued to evolve. Pulse lengths increased from a fraction of a second to much longer time periods. In the past it was sufficient to wait until after the device pulsed to begin collecting data and making it available to the scientists. As pulse lengths extended and MDSplus was used to collect data for extended time ranges it was necessary to add features to record data continuously while making the data accessible to the users. In 2007, long pulse extensions were added to MDSplus to handle this new requirement. [7]

The MDSplus core development team has changed over the years in both size and personnel. Except for a two year period where several laboratories provided additional support for the porting of MDSplus to other operating system, the team that continued to development and support MDSplus has consisted typically of 3 or 4 people working part time on MDSplus while providing local computer support of the experiment operation at their laboratories. The bulk of the time the MDSplus effort was focused on adding new features as needed and providing support to the user community by assisting with the installation and configuration of MDSplus and fixing bugs in the code as needed. More recently more effort has been concentrated on quality improvement of the MDSplus software utilizing modern development tools for discovering problems that may cause memory leaks, race conditions and potential references of uninitialized memory which could cause unpredictable results.

Much like MDS, news of MDSplus spread to the rest of the fusion energy research community. Not long after its first use at MIT and Padova, MDSplus started appearing at fusion research facilities around the world.

3 MDSplus Today

Since its first use in 1991, the functionality of MDSplus has grown considerably. MDSplus originally developed for the OpenVMS computer operating system[8] has since been ported to many different platforms and now there are installation kits and repositories available for about 15 variations of Linux systems as well as MacOSX and Windows. MDSplus now provides interfaces to many programming languages and third party applications giving the scientists and engineers a wide choice of tools they can use to view or manipulate the scientific data. Modern object oriented based tools and compilers can now take advantage of the object oriented nature of MDSplus designed into the system from its beginning. An advanced automated build system[9] now tests and generates new MDSplus releases nightly using state of the art tools such as Jenkins[10], Docker[11], and GIT[12].

Initially developed for short pulse fusion experiments, MDSplus now provides high performance features for recording large quantities of streaming measurements which at the same time can be actively monitored by the scientists and engineers. Today, the typical quantity of data recorded during each fusion experiment cycle is more than 1000 times what it was when MDSplus was first developed. In order to handle efficient data access for the very large signals produced in the most recent long discharge experiments, MDSplus provides support for the definition of dynamic, user specific Region of Interest (ROI) that define a time interval and possibly a resampling level for all the operations following the ROI setting. When large signals are accessed, only the signal portions corresponding to the current ROI are accessed. Since this operation is carried out at the lowest data access layers, large signal management is efficiently performed with a limited use of resources. For example, when accessing large signals stored on disk, only the sections containing useful data will be accessed. Currently, signals composed of several billions samples are routinely managed in some MDSplus installations.

While the functionality of MDSplus has grown significantly since the first pulses of the C-Mod and RFX experiments of the 1990's, much care went into making sure that the original data stored and the results of mathematical expressions stored with that data produce the same results now as they did back then. Great effort has also been put in guaranteeing backward compatibility not only in data format but also in software interfaces. Exceptions to this rule have been new component releases that required some iterations with users before reaching a stable interface. For this reason two MDSplus distributions are available: stable version, where backward compatibility is guaranteed as far as possible, and the "bleeding edge" alpha distribution that includes the latest features, but is normally not intended to be put in production systems.

An important aspect of every software package is its documentation. Often open source projects lack in introductory and user friendly documentation, and in the past MDSplus was no exception. For this reason, the MDSplus documentation available at the MDSplus web site (www.mdspus.org) has been improved and a tutorial section with working examples has been added.

Today, MDSplus is used at over 40 fusion research sites. It is open source and readily available over the Internet and analytics on the MDSplus web site shows it is also being used in many fields unrelated to fusion research including space exploration.

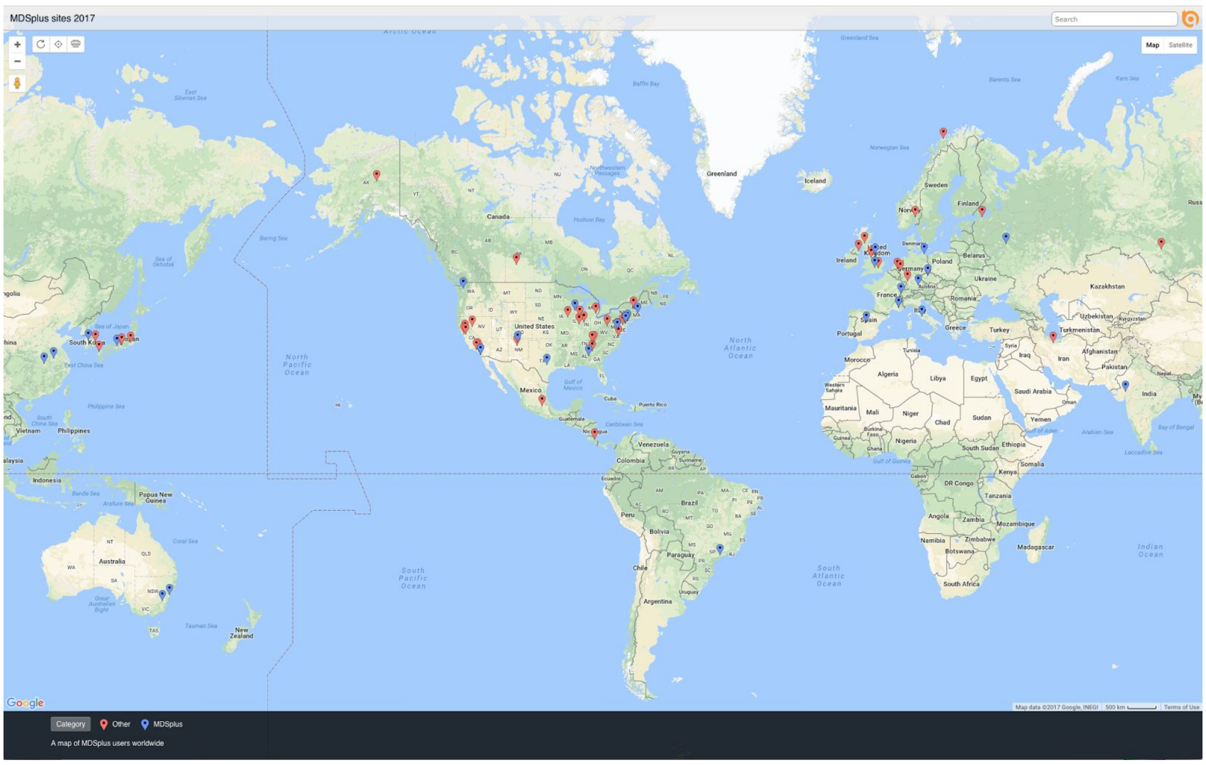

Figure 1. Map of Magnetic Fusion Experiments. Blue markers for MDSplus sites.[13]

4 MDSplus Tomorrow

The number of research facilities using MDSplus continues to grow each year and this growth is expected to continue for the foreseeable future. In order to cope with the rapid evolution of information technology, preserving at the same time the rugged concepts of MDSplus, work is ongoing to make the system more modular and to extensively support plug-in components. This has already been carried out for remote data access where the mdsip protocol, originally built over TCP/IP, can now be integrated with other network protocols, provided they are embedded in a component with a given interface. SSH, HTTP, UDT protocols have already been integrated and other existing and future protocols can be integrated as well. A similar concept is being worked out in order to decouple disk access, currently using POSIX interfaces, and to define a data access interface that can be implemented using different strategies such as cloud storage or other big data technologies.

There have been concerns expressed from the user community about how much longer MDSplus will continue to be supported and enhanced. Some of the original core development team have retired and the three remaining developers of that original team are not getting any younger. Is there any hope for the future of MDSplus? The answer is definitely ‘yes’, for several reasons, the most important of which is that the core development team has doubled in size over the last few years. The author list of this paper lists the current core development team. The younger team components brought new ideas and knowledge of modern development tools and strategies as well as great energy and enthusiasm for improving the quality and feature set of MDSplus. In addition to new team members, the change to using git and github for source code management along with automated testing enables developers from the user community to report MDSplus problems, suggest bug fixes and even submit code to fix or enhance MDSplus to be reviewed and merged into the MDSplus code base.

5 Conclusions

MDSplus has been around for quite a very long time and is used by many in fusion energy research worldwide. It has continued to evolve and keep pace with both the changes in technology and the functional requirements of the user community. With the addition of new core developers with knowledge of modern software development technologies and the continued support of the fusion community and their funding sources there is no reason to be fearful that MDSplus will not continue to grow in functionality and be supported long into the future.

6 Acknowledgements

The success of MDSplus stems largely from ideas from the user community and their willingness to test new features and report problems when they occur. It would be difficult to list all the scientists and engineers from the user community who have contributed but their participation has been greatly appreciated and has help shape MDSplus to what it is today.

A portion of this work is being performed under US Department of Energy Award Number: DE-SC0012470.

References

[1] Stillerman, J. A., T. W. Fredian, K. A. Klare, and G. Manduchi. "MDSplus data acquisition system." *Review of Scientific Instruments* 68, no. 1 (1997): 939-942.
[2] MDSplus Introduction. Retrieved October 19, 2017 from http://www.mdsplus.org
[3] Fredian, Thomas W., and Joshua A. Stillerman. "MDS/MIT high-speed data-acquisition and analysis software system." *Review of Scientific Instruments* 57.8 (1986): 1907-1909.
[4] Mackay, Hugh, et al. "Reconfiguring the user: using rapid application development." *Social studies of science* 30.5 (2000): 737-757.
[5] SDLC - RAD Model. Retrieved October 19, 2017 from https://www.tutorialspoint.com/sdlc/sdlc_rad_model.htm
[6]Horne, Steve F., Martin Greenwald, Tom W. Fredian, Ian H. Hutchinson, Brian Louis Labombard, Josh Stillerman, Yuichi Takase et al. "Remote control of alcator C-mod from Lawrence Livermore National Laboratory." *Fusion Science and Technology* 32, no. 1 (1997): 152-160.
[7] Fredian, T., J. Stillerman, and G. Manduchi. "MDSplus extensions for long pulse experiments."

Fusion Engineering and Design 83.2 (2008): 317-320.
[8] OpenVMS. Retrieved October 19, 2017 from http://en.wikipedia.org/wiki/OpenVMS
[9] Fredian, T., J. Stillerman, and G. Manduchi. "MDSplus automated build and distribution system." *Fusion Engineering and Design* 89.5 (2014): 649-651.
[10]Berg, Alan Mark. *Jenkins Continuous Integration Cookbook*. Packt Publishing Ltd, 2015.
[11]Merkel, Dirk. "Docker: lightweight linux containers for consistent development and deployment." *Linux Journal* 2014.239 (2014): 2.
[12]Loeliger, Jon, and Matthew McCullough. *Version Control with Git: Powerful tools and techniques for collaborative software development*. " O'Reilly Media, Inc.", 2012.
[13] https://batchgeo.com/map/4cb145f00105157a242b5acafa8e646b, retrieved October 19, 2017.