Applied Mechanics and Materials
ISSN: 1662-7482, Vol. 885, pp 170-183
doi:10.4028/www.scientific.net/AMM.885.170
© 2018 Trans Tech Publications, Switzerland

Submitted: 2018-05-02
Revised: 2018-07-26
Accepted: 2018-08-06
Online: 2018-11-20

Methods and Technologies for Research- and Metadata Management in Collaborative Experimental Research

Nils Preuss[2,a], Georg Staudter[1,b], Moritz Weber[1,c], Reiner Anderl[1,d*], Peter Pelz[2,e*]

[1]TU Darmstadt, DiK, Germany

[2]TU Darmstadt, FST, Germany

[a]nils.preuss@fst.tu-darmstadt.de, [b]staudter@dik.tu-darmstadt.de, [c]m.weber@dik.tu-darmstadt.de, [d]anderl@dik.tu-darmstadt.de, [e]peter.pelz@fst.tu-darmstadt.de

Keywords: research data management, metadata management, documentation of experiments

Abstract. Newly developed technologies and methods for the purpose of controlling uncertainty in technical systems must be proven and validated against reliable experimental studies. The availability of descriptive metadata is mandatory to enable long term usability and sharing of such experimental research data. This article introduces a concept for a software independent solution for managing data in collaborative research environments. The proposed approach leverages the advantages of capturing metadata in a uniform, modular data structure and providing software independent access to a centralized data repository as well as its contents by means of a web-application. The article presents a prototype implementation of the proposed approach and discusses its application on the demonstrator test rig of a collaborative research centre in the field of mechanical engineering.

1. Introduction

Collaborative research groups invest a considerable amount of time, know-how and money conducting experimental studies. Therefore, facilitating the sharing and reuse of data as well as avoiding redundant studies are key factors for success. However, there is always the risk of decaying information value over time. The phenomenon of “information entropy” can be caused by a variety of reasons, which is qualitatively illustrated in Figure 1, based on [1].

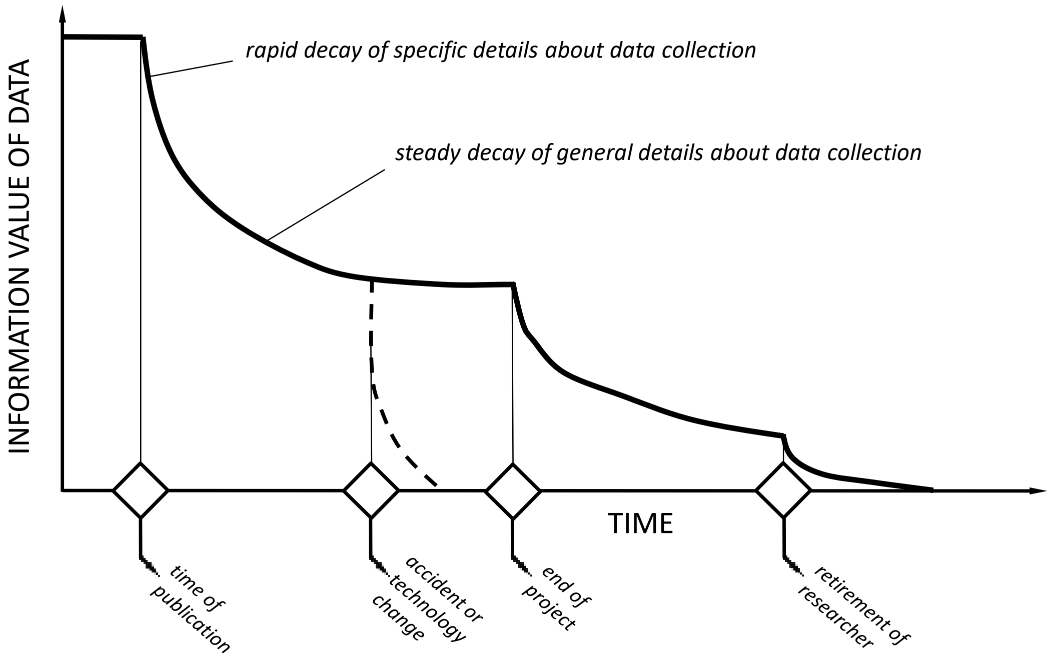


Figure 1: Information value of data over time

At the time of the research project at which data was generated, details about the development of datasets as well as information about the process are easily recalled and present in the memory of the scientist. Over time, memory begins to fade and, at the latest when the responsible scientist leaves the company, or in the event of a technological breakdown or -change, a significant amount of information is lost or unusable [1]. This is aggravated by the fact that datasets from simulation and experiments are still often stored locally, software tools for data analysis vary among the scientist involved and information about the creation of datasets are not sufficiently documented resulting in data that is not permanently accessible, interpretable, traceable and verifiable.

To counteract this decay of information value, meeting the following three requirements must be ensured: Firstly, data is stored in a repository that is accessible to all researches within the collaborative research group at any time. Secondly, data is stored in a uniform format in order to guarantee access to its content. Thirdly, any data collection contains all relevant metadata to guarantee its interpretability. The availability of descriptive metadata as well as a reliable and convenient access to archived data is mandatory for facilitating data sharing and reuse and thus for the long term usability of the generated experimental research data [2] [3] [4]. Only with existing descriptive metadata, data evaluations of any form are permanently traceable and verifiable [5] [6]. This is increasingly important facing the highly heterogeneous experimental setups and research objectives typically found in a collaborative research environment [7]. Additional individual metadata is implied in the phases of development (e.g. construction or simulations), manufacturing (e.g. control data of machining processes) and usage (experimental data). Knowledge about relevant metadata is a baseline requirement to describe, evaluate und master the remaining uncertainty in data management. Especially in collaborative research, uniform and software independent data access is needed to provide equal benefit for all researches and participants [8] [9]. The goal is to manage datasets by capturing metadata in a uniform data structure and to centralize its access to improve the overall level of documentation, to allow the traceability of results and to reduce the barriers associated with collaboratively used data.

This work focuses on the requirements typical for experimental research in the domain of mechanical engineering, specifically the major challenge of high heterogeneity across experimental setups mentioned above. The majority of test rigs in mechanical engineering are typically unique prototypes and instruments for measurements are highly customized. As a result, researchers produce data with highly heterogeneous structures. Existing standards for metadata and data structure are not suitable to describe the varying mixture of devices and methods employed for data acquisition and analysis sufficiently. For those reasons, the adoption of out-of-the-box software is difficult for this field of research and an applicable solution has to be developed.

Similarly, recent work in other research disciplines has developed efficient web-based solutions for data management in those domains, based on open-source software tools [10] [11] [12]. [13] gives a comprehensive overview of RDM approaches in other institutions and lists the relevant disciplines as well as corresponding literature. The cited author has implemented a research data management system for a collaborative research environment focusing on the management of (geo-) data produced within the field of earth and environmental sciences. While the developed metadata schema is customized for that specific domain, and therefore not directly applicable for the requirements of mechanical engineering, its basic system-architecture has proven to be successful [14] [15] [16], which is why this work follows a fundamentally similar architectural approach.

The future goal to provide public access to the generated research data will demand utilization of project-independent infrastructure provided by public service providers, such as the university computing centre. However, this article aims to present the general approach and early results of a prototypical solution for the documentation of experimental research data and centralized access to it, using infrastructure internally available for testing.

The modular active spring-damper system developed at the Collaborative Research Centre (CRC) 805 “Control of Uncertainties in Load-Carrying Structures in Mechanical Engineering” serves as an application example for the proposed concepts.

2. Approach

The overall goal is to provide a software independent solution for managing data especially in collaborative research environments, to unify the data structure and to centralize its storage. Furthermore, it is essential to provide an intuitive solution for users to find the data they are looking for and to identify related datasets. The approach aims to ensure that data retains its value of information over time and to ease the access for researchers who were not directly involved in the creation of the datasets. Therefore, the research is divided into three main areas: (i) ensure central and software independent data access, (ii) ensure content interpretation and information value by finding a uniform data structure with integrated metadata, (iii) ensure content access and interaction by providing a suitable solution for navigating, displaying and interacting with datasets. The following section addresses the three main functionalities derived from the introduction and presents possible methods and technologies for improvement.

2.1 File storage and access. Since the goal of collaborative research is the sharing of data, it is a fundamental requirement, that the data access is ensured from anywhere at any time for all participants. Unfortunately, it is still common that datasets are stored locally and that only its producer is able to not only access but also interpret the information. For many collaborative research institutions, the mere availability of a central storage of all their datasets in one single location would be a significant improvement in terms of data management. For this purpose, a wide range of database management systems (DBMS) is available, which build the basis for this approach.

A Database Management System is the software that is responsible for the management of stored data and that performs the access to the database requested by the user or application programs [17].

The basic functionality requirement for a data management system is the ability to store, locate and retrieve all data in its domain. Nowadays, this also allows users to have a web-based access to the files. Its basic architecture consists out of three tiers: Client, Application Server and Database Server, c.f. Figure 2.

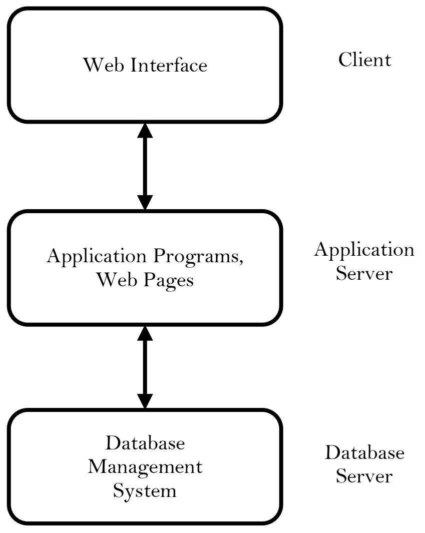


Figure 2: DBMS architecture

The web interface is used by the client to communicate with the server through the intermediate layer. This tier accepts requests from the client, processes the request and sends the commands to the database server. The database tier sends the result back to the client via the middle tier. To ensure data security, the DBMS protects the database from unauthorized access by asking for authentication via user name and passwords. Since one of our goals is to enable users to access files from anywhere and to keep the level of data security, DBMS remote extensions i.e. web-applications are a suitable solution for our approach. To implement custom made web-applications, DBMS typically provide Application Programming Interfaces (APIs). State of the art for web-APIs are Representational State Transfer (RESTful) versions of web services based on HTTP-based machine communication that transmits data in a variety of formats, such as plain text, Extensible Markup Language (XML), JavaScript Object Notation (JSON), or Stream [18]. By using this approach, we can provide a central storage of all datasets in one single location, access from anywhere at any time and guarantee the data security.

Within the CRC 805, the Alfresco® platform has already been set up as the central data management system. It is an open source platform, supports open standards and APIs for custom web-applications [19]. Using this platform for the purposes of this research project has the advantage, that its setup is well known to all participants within the collaborative research project, a significant amount of data and information are already located there and most importantly, that security and access administration for all research members is already existent.

2.2 Data interpretation and utilization. As the start of this section outlines, a major requirement for a data management solution in collaborative research is retaining the information value of generated data. This means ensuring that any researcher in the research environment can find, interpret and utilize relevant data collections. The research workflow, utilized equipment and software as well as the structure of datasets produced within a collaborative research group is highly heterogeneous. Therefore, a uniform documentation of descriptive information is key to enable collaborative interpretation and utilization of data collections.

To interpret any given dataset, its documentation must be directly available, which can be achieved by including it in the data collection and linking it with the respective datasets. Furthermore, to utilize this documentation by means of software tools, it must be available as structured, machine-actionable metadata. It is therefore necessary to define a uniform nomenclature and data model to represent a wide range of research tools and workflows, in the context of this work experimental setups and procedures.

The approach proposed in this article is to specify the data model piecewise by defining classes of metadata objects for each specific case of the information aspects required for interpretation and utilization. It is critical to include representations of the following information in the data model representing documentation of an experiment [20]:

(i) Geometric, configuration and operating parameters of the system under test. Documenting the appearance and construction of a test rig is comfortably handled by CAD-software and associated established data models. However, summaries of characteristic geometric parameters should be readily available outside of CAD-data because they enable identifying data collections of interest and may be relevant to analysis of the contained datasets. The same applies for configuration and operating parameters that are not stored as distinct datasets within the collection. This information includes descriptions, quantities and units. It is encapsulated by objects of class *parameter*.

(ii) Auxiliary information describing properties and characteristics of utilized equipment. Knowledge of equipment properties is needed to reduce uncertainty when identifying measured quantities of an experiment and investigating the plausibility of results. Programmatically available equipment characteristics are needed for post-processing of raw data. It is imperative that this information is available in the data collection and not implied or hidden in software configurations, comments or code responsible for data generation or processing. This is especially important if frequent changes of utilized equipment or recalibrations are necessary. This information is carried by objects of class *instrument*.

(iii) Information describing contents and origin of a dataset. Contents of a dataset might for example be a time series, raster data or a set of statistical values. Each type must be described via respectively characteristic properties to enable definite interpretation. A time series dataset is characterized by the time interval between samples among other things. For a raster dataset the resolution or orientation might be relevant. At the very least, a timestamp should be attached to each dataset regardless of its type. This information is held by objects of class *dataset*.

On the basis of these classes, it is possible to specify how metadata objects may be combined to a valid metadata structure. To ensure traceability of any datasets origin, information relating data to the equipment it was produced by is needed. To reflect this, an object of class *pipeline* holds any number of *dataset* and *instrument* objects, relating them to each other. Additionally, in the case of datasets derived by means of data processing, the source dataset must be known. Input data is related by specifying *pipeline* objects to reference other *pipeline* objects, holding the source *dataset* objects. Those relations of equipment, data and parameters are of course only valid within a certain scope. To define this scope, all *pipeline* and *parameter* objects belonging together are held by an object of class *measurement run*. It is reasonably implied that the setup of equipment and system under test is not changed over the course of a single measurement run. Figure 3 shows the class based concept introduced above in UML-notation.

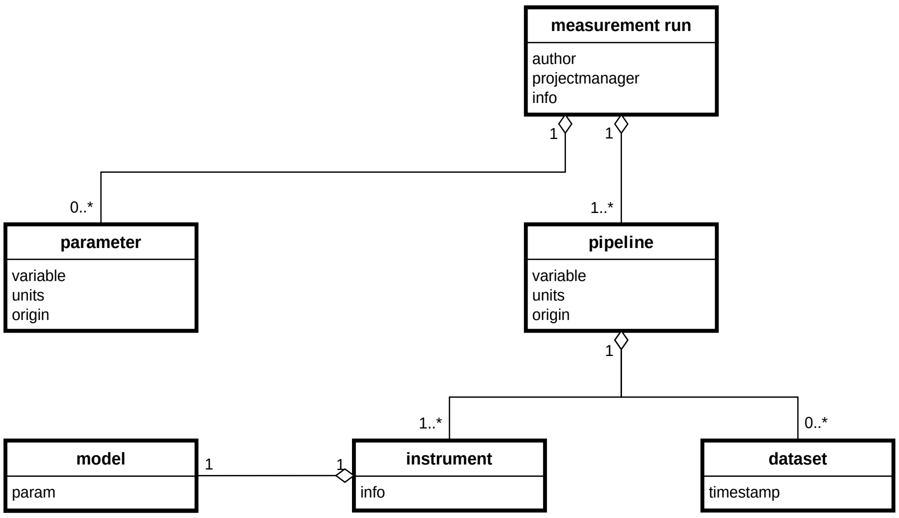


Figure 3: UML diagram showing the data model, forming a hierarchic structure of generic classes

2.3 Data access and interaction. Today, another critical factor to facilitate working with shared data within collaborative research environments is the need for tools that allow collaborators to access the data content and interact with complex datasets of varying provenance and format. The current situation is, that software tools for data analysis are closely related to the field of scientific research, that the different tools work with different data formats and therefore collaborators are required to install and understand/learn a variety of software tools to access and interact with content. This requires a significant amount of time and negatively affects the overall scientific productivity.

Our approach is to provide a software- and platform-independent tool that uses a web-application based solution to display and interact with data content externally, which eliminates the need to install specialized software and the use of special operating systems. For this purpose, we use the RESTful API provided by Alfresco and references to data objects (ID) to access the Alfresco Repository and to display the contents on the web browser, c.f. Figure 4.

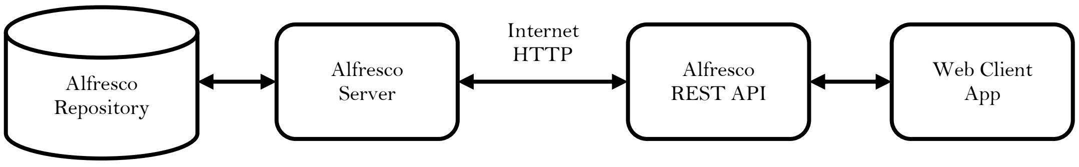


Figure 4: Architecture web application

The application sends HTTP requests to the Alfresco server and data is accessed by sending a URL using one of five HTTP API methods: GET (select), PUT (update), POST (insert) and DELETE as well as OPTION methods. These procedures all act on a route with a ID that represents a data instance, or in case of POST, create the ID of a resource. The server responds to the request with a HTTP response and a JSON object, which can be visualized in the browser.

Visualization of content as well as solutions for exploring the relations among datasets are essential for a productive scientific environment. The majority of web browsers today typically support a variety of file formats and can render a HTML markup into a human friendly presentation. Most formats, which are not supported by the initial browser setup, can usually be visualized by downloading and adding a suitable plugin to your browser. In case of research data from experiments and simulation it is not guaranteed that a corresponding plugin already exist, which is a critical factor for the accessibility of the content. In addition, our aim is not just to deploy a web-interface for users to access and view the content of data but to find datasets and identify related files. The data model described in the previous section in combination with a self-descriptive file format prevents requiring the client side to have detailed information about the exact structure of the data files.

To implement the data model presented in section 2.2, the underlying file format must be able to represent data structures that include complex relations of data and metadata. Therefore, a file format implementing a graph based data model is preferred, which allows relating data or metadata records directly. These relations can be marked up from text strings following a special syntax, using formats like XML or JSON, which are widely accepted standards. However, other state of the art file formats can easily be serialized to XML or JSON while meeting the following additional requirements:

An especially important criterion is (i) the ease of use, as the developed solution should interfere as little as possible with active research. A researcher must be able to utilize the file format in any programming language and operating system of choice through a simple interface, ideally with a moderate learning curve. This in turn implies a high degree of (ii) portability and or interoperability, as well as (iii) language and platform independence. Extraction of data subsets must be fast and uncomplicated to implement, thus (iv) random access of contents is desired. To ensure long term accessibility of contents as well as convenience, it is important that (v) the file format is self-descriptive. This means the format includes all necessary information about data types and dimensions to read and reconstruct the contents of a file. The user, application or program does not need to know the implemented structure beforehand, it can instead be programmatically explored.

HDF5 is an open source, general purpose file format that provides the foundation to structure data on disk as a directed acyclic graph (DAG) and to attach lists of attributes to each data object. The File format is independent of storage medium, programming environment and operating system. Open source platform independent bindings to read and write HDF5 files are maintained by the HDF group for C, C++, CLI (.NET), FORTRAN and Java. As HDF5 has been adopted by a large community over a broad range of disciplines [21], third party libraries to operate on HDF5 files exist for all programming or scripting languages relevant in the scientific community. Examples include MATALB, LabVIEW, Python, R, Mathematica, IDL, Julia and Octave [22].

Figure 5 shows the abstract base model of HDF5 that allows users to form hierarchic or DAG structures from a set of base objects [23]. These include the file root, groups and datasets, all of which can be further described by a set of attributes as key-value pairs. Each object is uniquely identifiable and accessible via file internal URLs, allowing for random access of file contents. The stored representation is self-describing, which allows to view and edit any file content in the available graphical editor [24], independent of actual stored data or structure. Additionally, tools for the conversion of HDF5 to data exchange formats like XML or JSON are available [25] [26].

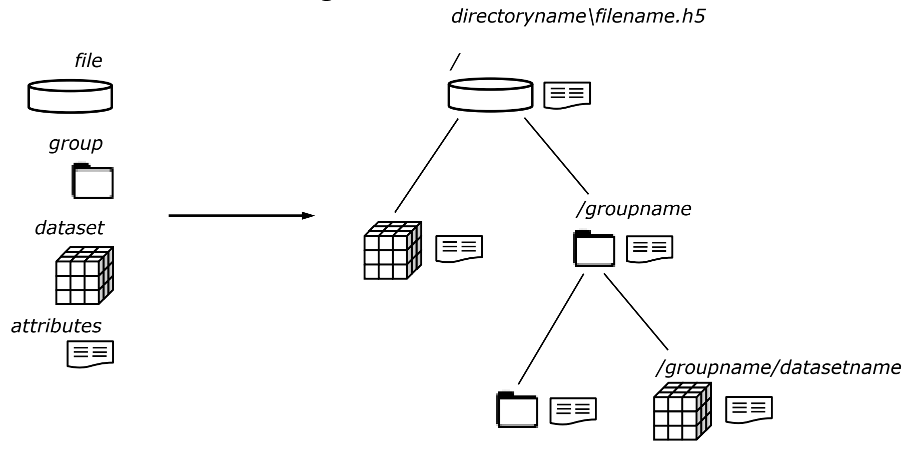


Figure 5: Overview of HDF5 base model objects and exemplary hierarchic data structure

These above described features lead to a high level of interoperability, which motivated the decision to use HDF5 over other formats designed to store experimental data [27]. Various projects and communities similarly use HDF5 as the foundation for more domain specific data models [28] [29] [30] [31]. The utilization of HDF5 enables the development of a flexible data model at an excellent cost-benefit-ratio, while still serving as a stepping-stone to other file formats or database solutions.

3. Prototype Implementation

According to the approach outlined in this paper, the web application prototype was implemented using the architecture shown in Figure 6 and Alfresco's RESTful API. The web applications frontend was designed using Twitter Bootstrap for styling as well as JavaScript libraries such as Jquery for event handling. Bootstrap is a front-end library for designing websites and web applications via standardized HTML that can work on many types of devices and browsers. It contains HTML- and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as JavaScript extensions. Unlike many other web frameworks Bootstrap hides the complexity of JavaScript, which makes development more efficient. Although the current implementation serves only the basic functionalities, the layout was designed in accordance with ISO 9241, a multi-part standard covering ergonomics of human-computer interaction, to ensure an intuitive handling. Figure 3 shows the conceptual lauyout of the application prototype [32].

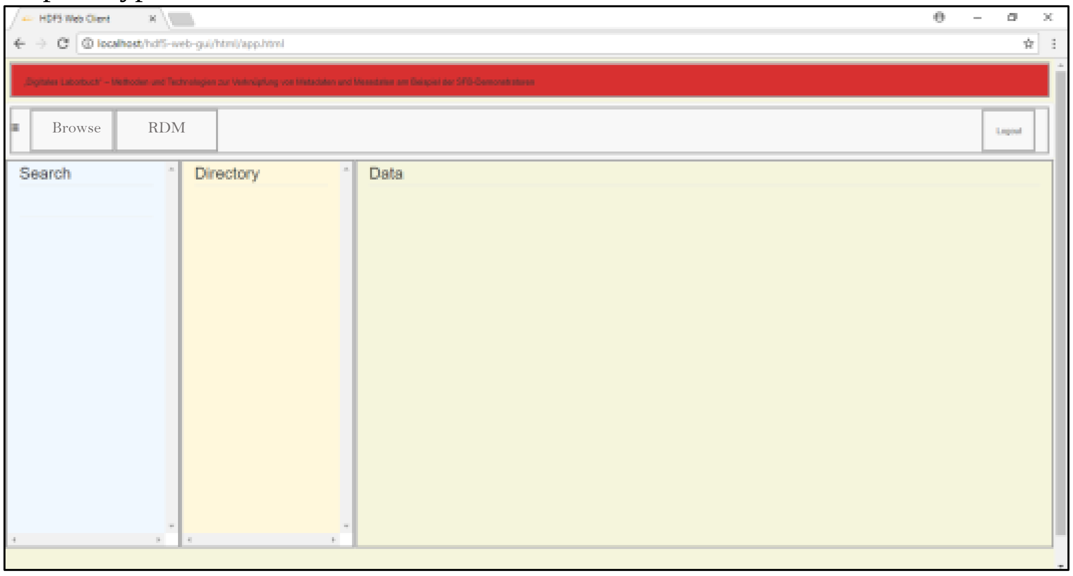


Figure 6: Layout draft of the prototype

When browsing to the application URL, users have to login using their Alfresco login credentials in order to authenticate and gain access to the central file storage location. A successful authentication gives the user a familiar view, inspired by the Alfresco web client page, that provides the folders and content stored on the Alfresco server and to which he/she is authorized to access. Not only does this ensure our goal to maintain the high level of security, but also to provide access to the files produced in the earlier periods of the CRC and to present them in a familiar and thus easily adoptable way. Files can either be downloaded or viewed in the browser (png/jpg and pdf). By clicking on the RDM-button (Research Data Management) users will be guided to the application main functionalities, which are addressed in this paper. In order to visualize the content as well as the hierarchical tree structure of the HDF5 files h5serv by The HDF Group was implemented and hosted on the Alfresco server. h5serv is a Python-based web service that is used to access HDF5 data over the web using its RESTful API, where each HDF5 data object or entity is identified via a Uniform Resource Identifier (URI) [33]. JSON is used as the default representation for requests and response. This way, HDF5 datasets are stored in a central location and proceeded via URIs, which minimizes the number of bytes to be transmitted over the network and eliminates the need of impractically downloading files in order to be able to access the content. Within the "Directory" window shown in figure 4**,** the hierarchical tree-structure of the HDF5 datasets is visualized. Users are able to expand the tree and select individual data via mouse click. Subsequently, content such as text or tables are displayed in the window on the right hand side. In order to enhance the overall level of productivity and to minimize the time exposure of finding certain datasets and identifying related datasets a search function was implemented and placed on the left hand side of the UI.

To provide researchers with a convenient way to create data collections compliant to the uniform data model for experimental data described in section 2.2, an interface toolbox is developed. While users can of course use the native HDF5 API or the file editor to attach metadata to their datasets, it imposes a great effort to interface with large metadata structures in this manual way. Developing an API toolbox is chosen over a GUI tool because of the high degree of heterogeneity of data as well as the reusability of metadata creation scripts. The prototype API is implemented in MATLAB, which is the programming environment most widely used for dataset manipulation within the CRC.

The first primary objective of the API is to let the user conveniently create objects of the classes defined in section 2.2 and combine them into a hierarchical object structure like specified by the data model. This metadata structure of objects is written as a HDF5 file by storing object properties as attributes of HDF5 nodes (groups or datasets) and nested objects as nested HDF5 groups or datasets. The second primary goal of the API is to minimize code duplication while keeping the data model extensible. To achieve this, the interface implementation introduces an abstract class, whose capabilities are inherited by the classes that carry the actual information. This class implements methods for the organization of HDF5 URLs as well as reading and writing of attributes and nested objects.

The proposed metadata structure represented by the graph structure inside an HDF5 file is leveraged to enable the user to find files, data collections or datasets of interest in this database of HDF5 files. As explained above, a node (group or dataset) contained in an HDF5 file may represent an object of the data model, in this case its attributes represent the object properties. Investigating the nodes can easily be accomplished by performing a depth-first search by means of a recursive file parser using the native HDF5 API.

The user may specify the requirements for wanted files or contents in the following way: Wanted objects can be specified by listing the desired combination of attribute names and values of a node in the HDF5 file structure. If multiple different kinds of objects are required, this can be done by specifying multiple attribute lists. To verify the objects of interest, exist at some location in a file, the parser compares the attributes of every node to every specified list. If all the attribute names and values match, the location of the node is added to the results for the respective attribute list.

After the parsing is complete, the results are processed and presented to the user. This depends on the user's choice of rules which objects should be returned and how the different desired objects must appear in the file: (i) Any of the objects specified via an attribute list exists in the file. (ii) Multiple objects specified via different attribute lists exist in the file. (iii) Multiple objects specified via different attribute lists exist in the file and are connected to each other. The latter is confirmed by comparing the locations of the results and verifying that one object URL includes the other. Results not obeying the chosen rules are removed. A list of the file and node location for all remaining results are presented to the user, if the respective attribute list has been flagged for return. Clicking on an entry in the list of results takes the user directly to the corresponding file and object location, using the data view window of the web-application. Figure 7 illustrates the described concepts. The next section gives a concrete application example.

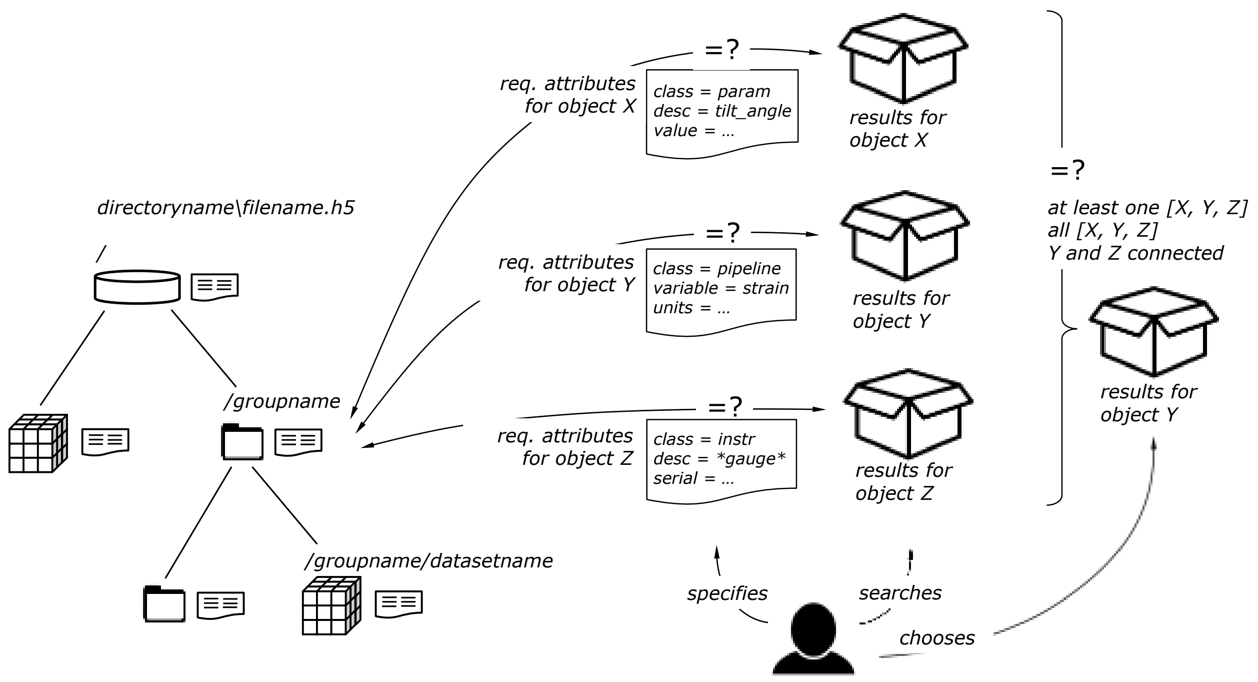


Figure 7: Working principle of the parser used for data and metadata search requests

4. Application Example

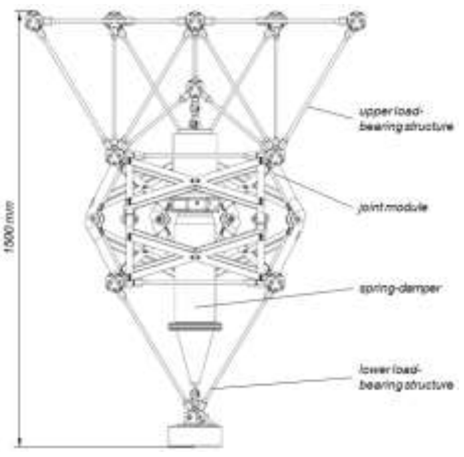


Figure 8: Demonstrator test rig of the CRC 805, or MASDS

To test the approach and prototype implementation described in the previous sections, the demonstrator test rig developed within the CRC 805 serves as a suitable application example. The CRC 805 demonstrator, or MASDS (Modular Active Spring-Damper System), is a central platform across all subprojects within the CRC to validate methods and technologies for controlling uncertainty in load-carrying structures. This includes characterizing, quantifying and assessing uncertainty in regard to load distribution, stability and vibration control. The MASDS essentially consists of two load-bearing structures of beams, connected via a joint module and a semi-active spring-damper, cf. Figure 8. Equipped with a variety of sensors, the MASDS is a test rig for a significant number of experiments, such as drop impact tests. The operating parameters, as well as some of the geometric parameters of the demonstrator test rig, are highly dependent on the respective experiment and the corresponding objective of the subproject. At the same time, which of the available sensors are utilized may vary and additional instrumentation may be introduced to the system when testing a prototype of a technology developed in a subproject.

Currently, for each experiment series, a new directory on the local file system is created. The measurement data for each experiment is stored to a MATLAB file: Generated datasets corresponding to the measured signals are stored to separate variables. At this point, signals, or rather responsible instruments and their positioning in the test rig can only be identified by means of the abbreviations used in the file- and variable-names. Information about a series of experiments, i.e. varied parameters and enumeration of measurements, if at all, is only available in the form of an unstandardized readme txt, MS excel or pdf file. Similar situations are prevalent for experimental research in the domain of mechanical engineering, especially in academia [34]. As a result, identifying data that meets specific requirements and all auxiliary information required for its interpretation or utilization is cumbersome and time consuming. In many cases, the involvement of multiple persons that have scattered knowledge regarding needed details is required.

The following section shows the application of the proposed approach on an experiment series conducted at the MASDS, as well as the gained advantages, using the presented implementation prototype. For illustration purposes, the use case of a researcher looking for data generated at the lower load bearing structure, more specifically at the beams, is considered. In the following, only the representation of information relevant to this use case is described. However, the remaining aspects of the test rig are represented similarly. The data and metadata relevant for the use case includes: (i) the load mass, tilt angle and drop height of the MASDS as operating parameters, (ii) the normal and bending strain of the beams, i.e. properties of the corresponding strain gauges as well as (iii) measured time series including timestamp and units.

Following the data model introduced in section 2.2, the information regarding (i) is represented by creating objects of class *parameter*. The representation of (ii) is two-fold: On the one hand, one object of class *pipeline* per measured strain is created. On the other hand, one object of class *instrument* representing each respective strain gauge is added to the corresponding *pipeline* object. Representing (iii) is achieved by adding one object of class *dataset* per measured time series to the responsible *pipeline* object. This relates the respective *dataset* and *instrument* objects to each other. The *pipeline* and *parameter* objects are added to an object of class measurement run, which represents the experiment. Figure 9 illustrates the HDF5 data structure resulting from writing this metadata object structure to a file. The data which the researcher is looking for is highlighted.

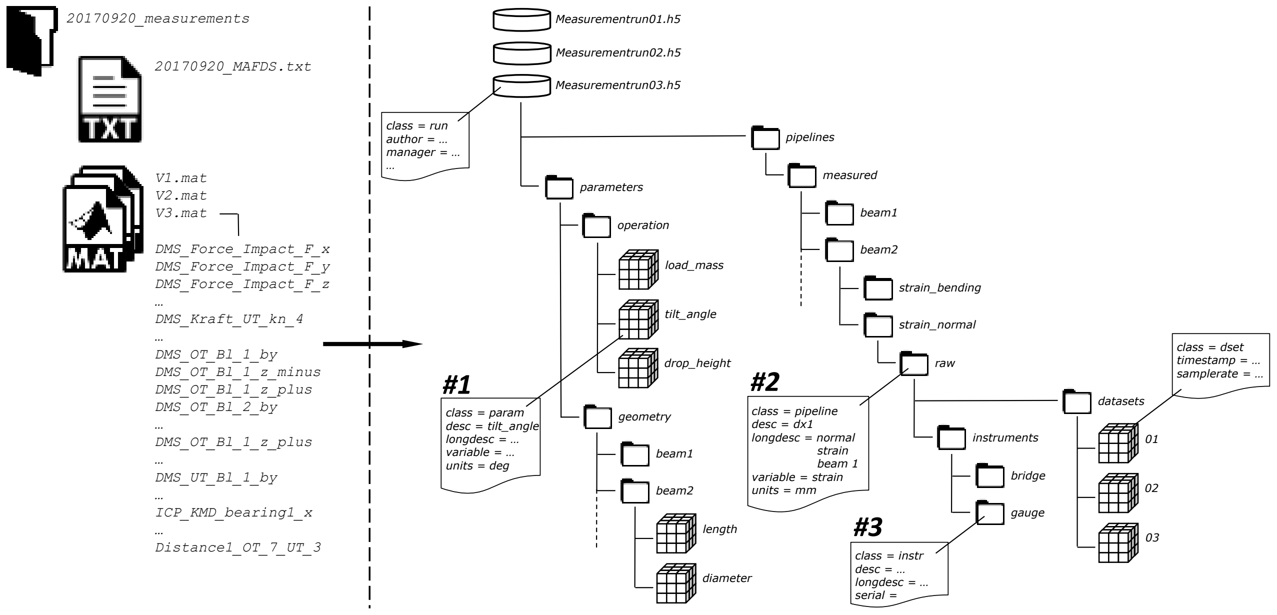


Figure 9: Exemplary HDF5 data structure for a drop test experiment at the MASDS

Using the web app, the researcher has the following options to find the data of interest. On the one hand, the user has the conventional option of browsing the folders within the Alfresco repository, to download the files of interest and to view them via a corresponding viewer plugin. This may be a sufficiently effective way to find documents such as text files, but in the light of this application example it does not meet the goals outlined in this paper. On the other hand, the user can use the applications search function described in section 3. Using the GUI, the researcher states the required contents of available HDF5 files, as well as the node locations to be returned. For this purpose, a list of desired key - value pairs can be specified using the search GUI. This application example considers the following search (cf. Figure 7): (i) requested result is are strain measurements (class = pipeline, variable = strain), given that the related instrument is (ii) a specific gauge, specified by serial number (class = instrument, serial = #value) and (iii) a specific value of the tilt angle of the MASDS (class = parameter, value = #value). As described in section 3, clicking on an entry in the list of results takes the user directly to the corresponding file and object location, and the content is visualized in the data view window of the web-application, cf. Figure 10.

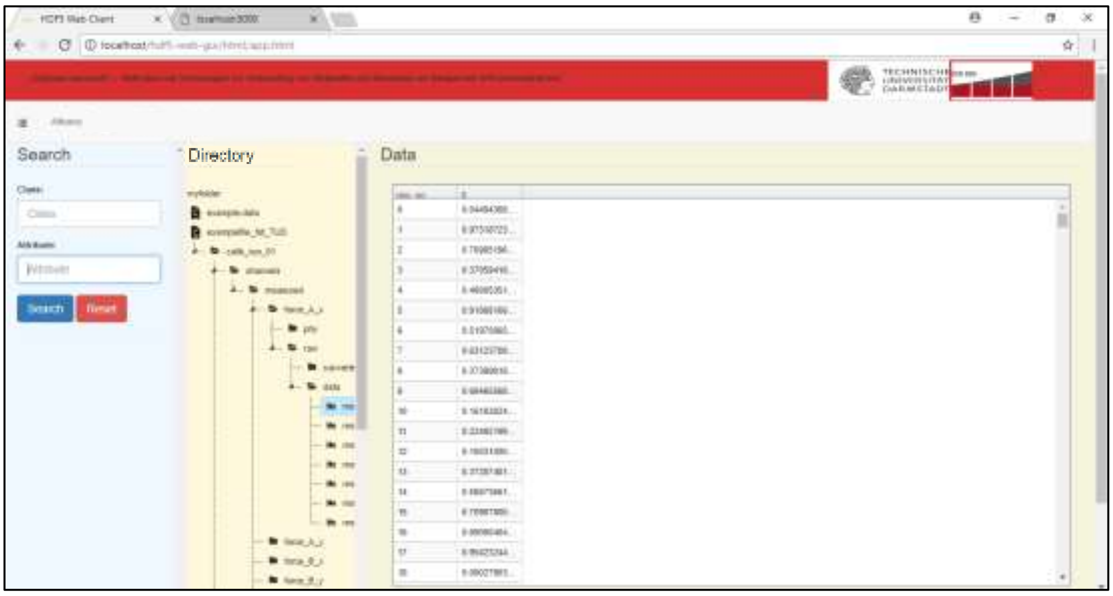
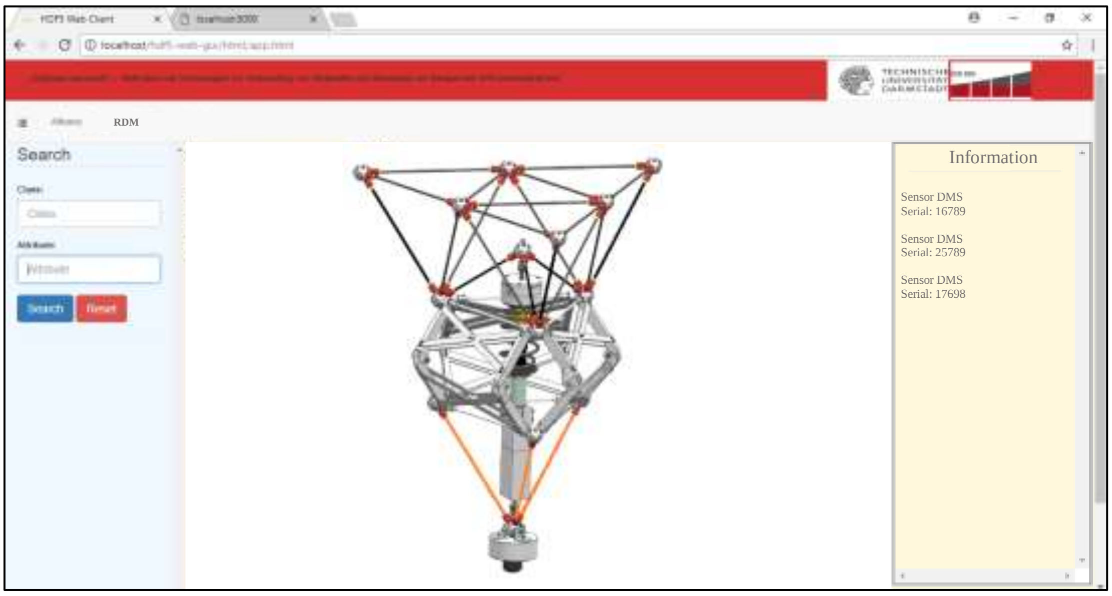


Figure 10: Example search result visualized in application



Information

Sensor DMS
Serial: 16789

Sensor DMS
Serial: 25789

Sensor DMS
Serial: 17698

Figure 11: Application with embedded 3D model of the MASDS

The researcher might be interested in looking for data generated at the lower load bearing structure, more specifically at the beams, but does not have sufficient information about installed sensors or measured quantities. For this purpose, the application provides a 3D-Model of the MASDS, providing sensor information. It can be displayed by clicking on the parts of interest, cf. Figure 11. This information, for example serial number and type of an installed sensor, can be added to the key - value list to execute the search. The 3D-Modell allows users to pan and rotate the assembly as well as to zoom to the parts they are interested in. The HTML-version of the MASDS was produced by exporting its 3D-CAD model from Siemens NX into JT (Jupiter Tesselation) format and converting it to HTML by using the Siemens JT2Go Desktop application.

5. Discussion and Conclusion

Facilitating the sharing and reuse of data as well as avoiding redundant studies are key factors for success within a collaborative research environment. This is typically limited by data being inaccessible for a large part of the research group as well as data being unintelligible for researchers not involved with the corresponding experiment. The presented approach of capturing metadata in a uniform data structure and implementing a data repository that provides uniform and centralized data access via a web-application enables users to find the data they are looking for and to identify related datasets and thus, solves the main points of improvement derived from the introduction. While previously datasets had to be manually found, then downloaded and visualized with appropriate software, the datasets can now be accessed and viewed using a standard web-browser.

Measurement data can be linked to extensive descriptive information with relatively low effort, facilitating the collective long-term usability of heterogeneous datasets. Ease of use is achieved as a result of the pure hierarchic data model and its file representation, keeping the complexity low. HDF5 as the underlying file format and the corresponding libraries provide the capabilities to explore the file contents programmatically as well as random access to the file contents. The software interface allows integrating the creation of appropriate data structures and metadata into the data acquisition software of the MASDS in the near future.

Future work will include improving the user interface and display views in order to improve the usability of the application, based on feedback from early adopters. Furthermore, opportunities to add additional features, such as tools for collaborative project management, will be evaluated. To unify data management across different research projects, a standard vocabulary and basic information requirements must be established, extending the data model and interface library accordingly.

Acknowledgements

The authors would like to thank the German Research Foundation (DFG) for funding this research within the Collaborative Research Centre (SFB) 805 ‘Control of uncertainty in load carrying structures in mechanical engineering’ (TU Darmstadt), spokesperson Prof. Dr.-Ing. Peter F. Pelz.

References

[1] W. K. Michener, "Meta-information concepts for ecological data management," (in English), *Ecological Informatics,* vol. 1, no. 1, pp. 3-7, Jan 2006.

[2] R. Van Noorden, "Data-sharing: everything on display," *Nature,* vol. 500, 2013.

[3] J. Greenberg, S. Swauger, and E. Feinstein, "Metadata Capital in a Data Repository," *International Conference on Dublin Core and Metadata Applications; DC-2013--The Lisbon Proceedings,* 09/02/ 2013.

[4] J. Ludwig, *Leitfaden zum Forschungsdaten-Management*. 2013.

[5] Deutsche Forschungsgemeinschaft e.v, "Leitlinien zum Umgang mit Forschungsdaten," 2015.

[6] M. Greenwald, T. Fredian, D. Schissel, and J. Stillerman, "A metadata catalog for organization and systemization of fusion simulation data," (in English), *Fusion Engineering and Design,* vol. 87, no. 12, pp. 2205-2208, Dec 2012.

[7] Z. Chen, D. Wu, J. Lu, and Y. Chen, "Metadata-based Information Resource Integration for Research Management," *Procedia Computer Science,* vol. 17, pp. 54-61, 2013/01/01/ 2013.

[8] W. Benger, "On Safari in the File Format Jungle-Why Can't You Visualize My Data?," (in English), *Computing in Science & Engineering,* vol. 11, no. 6, pp. 98-102, Dec 2009.

[9] D. P. Schissel *et al.*, "Automated metadata, provenance cataloging and navigable interfaces: Ensuring the usefulness of extreme-scale data," (in English), *Fusion Engineering and Design,* vol. 89, no. 5, pp. 745-749, May 2014.

[10] K. Nadrowski *et al.*, "Harmonizing, annotating and sharing data in biodiversity–ecosystem functioning research," *Methods in Ecology and Evolution,* vol. 4, no. 2, pp. 201-205, 2013/02/01 2012.

[11] C. Willmes, D. Kürner, and G. Bareth, "Building Research Data Management Infrastructure using Open Source Software," *Transactions in GIS,* vol. 18, no. 4, pp. 496-509, 2014/08/01 2013.

[12] D. Heimann, J. Nieschulze, and B. König-Ries, "A flexible statistics web processing service - Added value for information systems for experiment data," in *Journal of Integrative Bioinformatics* vol. 7, ed, 2010, p. 17.

[13] C. Curdt, "Design and Implementation of a Research Data Management System: The CRC/TR32 Project Database (TR32DB)," PhD thesis, Universität zu Köln, 2014.

[14] F. Zander, S. Kralisch, C. Busch, and W.-A. Flügel, *Environmental data management with the River Basin Information System*. 2011.

[15] M. Diepenbroek, D. Hannes Bremen, and Grobe, *PANGAEA ® als vernetztes Verlags-und Bibliothekssystem für wissenschaftliche Daten*. 2018.

[16] T. Lotz, J. Nieschulze, J. Bendix, M. Dobbermann, and B. König-Ries, "Diverse or uniform? — Intercomparison of two major German project databases for interdisciplinary collaborative functional biodiversity research," *Ecological Informatics,* vol. 8, pp. 10-19, 2012/03/01/ 2012.

[17] R. Anderl, "Virtuelle Produktentwicklung B; Produktdatenmanagement," ed. Germany: Technische Universität Darmstadt, 2015.

[18] (2011, January 01). *SoftwareEngineering (2011): Difference between Web API and Web Service*. Available: http://softwareengineering.stackexchange.com/questions/38691/difference-betweenweb-api-and-web-service

[19] I. Alfresco Software. (2018, May 02). *Alfresco Documentation*. Available: https://docs.alfresco.com/

[20] R. McClatchey, Z. Kovacs, F. Estrella, J. M. L. Goff, L. Varga, and M. Zsenei, "The role of meta-objects and self-description in an engineering data warehouse," in *Database Engineering and Applications, 1999. IDEAS '99. International Symposium Proceedings*, 1999, pp. 342-350.

[21] The HDF Group. (2017, October 30). *HDF5 Users*. Available: https://support.hdfgroup.org/HDF5/users5.html

[22] The HDF Group. (2017, October 30). *Summary of Software Using HDF5*. Available: https://support.hdfgroup.org/products/hdf5_tools/SWSummarybyName.htm

[23] The HDF Group. (2017, October 30). *HDF5 User's Guide HDF5 Release 1.10.0*. Available: https://support.hdfgroup.org/HDF5/doc/UG/HDF5_Users_Guide-Responsive HTML5/index.html

[24] The HDF Group. (2017, October 30). *HDFView Homepage*. Available: https://support.hdfgroup.org/products/java/hdfview/

[25] The HDF Group. (2017, November 17). *HDF5 XML Information Page*. Available: https://support.hdfgroup.org/HDF5/XML/

[26] The HDF Group. (2017, November 17). *HDF5/JSON documentation*. Available: http://hdf5-json.readthedocs.io/en/latest/

[27] A. Pfeiffer, I. Bausch-Gall, and M. Otter, *Proposal for a Standard Time Series File Format in HDF5*. 2012, pp. 495-505.

[28] R. E. Ullman, "HDF-EOS, NASA's standard data product distribution format for the Earth Observing System data information system," in *Geoscience and Remote Sensing Symposium, 1999. IGARSS '99 Proceedings. IEEE 1999 International*, 1999, vol. 1, pp. 276-278 vol.1.

[29] A. Ingargiola, T. Laurence, R. Boutelle, S. Weiss, and X. Michalet, "Photon-HDF5: An Open File Format for Timestamp-Based Single-Molecule Fluorescence Experiments," *Biophysical Journal*, vol. 110, no. 1, pp. 26-33, 2016/01/05/ 2016.

[30] M. T. Dougherty *et al.*, "Unifying Biological Image Formats with HDF5," *Communications of the ACM*, vol. 52, no. 10, pp. 42-47, 2009.

[31] D. C. Price, B. R. Barsdell, and L. J. Greenhill, "HDFITS: Porting the FITS data model to HDF5," *Astronomy and Computing*, vol. 12, pp. 212-220, 9// 2015.

[32] (2018, May 02). *Getting Started - Bootstrap*. Available: https://getbootstrap.com/docs/3.3/getting-started/

[33] The HDF Group. (2016, May 02). *HDF Server Homepage*. Available: https://support.hdfgroup.org/projects/hdfserver/

[34] Digital preservation services development group, "Research Data File Formats and Digital Preservation - Final Report," The Open Science and Research Initiative (ATT), Febuary, 2017.