The *e!DAL* JAVA-API: Store, Share and Cite Primary Data in Life Sciences

Daniel Arend, Matthias Lange, Christian Colmsee, Steffen Flemming, Jinbo Chen, Uwe Scholz
Cytogenetics and Genome Analysis / Bioinformatics and Information Technology
Leibniz Institute of Plant Genetics and Crop Plant Research (IPK), D-06466 Stadt Seeland, OT Gatersleben, Germany
{*arendd,lange,colmsee,flemming,chenj,scholz*}*@ipk-gatersleben.de*

***Abstract*—**
Background: **Life sciences are data intensive. High throughput technologies, like "Next-Generation-Sequencing", "Phenotyping" and other -omics technologies produce a huge amount of primary data. In classic scientific publication process, primary data is usually aggregated to a number of paragraphs in a journal article and proven by figures, tables and supplementary material. So it is that the value of primary data, on which the scientific conclusions are based on, do not get proper attention. But, the responsible use and efficient availability of digital resources is an important factor in the nowadays "e-science" age. The increased attention in public as well as in the research community leads to novel strategies and concepts for primary data citation, which must be substantively, underpinned by enhancements to classic data management systems.**

Results: **We present the JAVA-based *e!DAL*-API, a comprehensive storage backend for primary data management. It stands for (*electronical Data Archive Library*) and implement a primary data storage infrastructure, but with an intuitive usability like a classical file system. Main features are version and meta data management, data citations, support for information retrieval, persistent identifiers and its easy integration into existing data frontends and information systems. The API has been designed and tested using experiences from several research projects and literature studies.**

Conclusions: **Primary data preservation in life sciences is accompanied by enhanced requirements to data management systems. *e!DAL* combines this novel arising requirements with features known from file systems, databases, content management and version control to one homogeneous storage system. It supports an embedded use in stand-alone JAVA software or in server mode a data repositories for collaborative, remote accessible data services. Thus, *e!DAL* is an efficient complement for data frontends, information systems and data management systems. The JAVA libraries, Maven artifacts, sample code, a show case demo, and the API-documentation can be downloaded from: http://projects.ipk-gatersleben.de/eDAL-Project**

I. Challenges in the E-Science Age

Data intensive sciences such as astrophysics, social sciences, life sciences and in particular bioinformatics are the driving forces towards the "e-science" age. One major input in analytical research processes are terabytes of primary data, which is produced by "high throughput" technologies, such as "Next-Generation-Sequencing" (NGS), mass spectroscopy or imaging methods. As David Roos remarks ten years ago, "we are swimming in a rapidly rising sea of data" and there is barely no way to "keep from drowning" [1]. To some extend this pessimistic view has not become reality. In case of life science databases, scientists apply search engines or information systems to acquire speedily the specific information they need [2]. This promising technology is effective for life science databases [3]. But this is just the tip of the "data iceberg" [4]. An efficient access to the unpublished, "invisible" data is still an open issue: Scientific staff is collecting a lot of data and reduce them down to a number of paragraphs of theories supported by some aggregated figures, tables and selected material. Usually, the authors add to their articles links to externally managed supplemental material. However, the older an article is, the lower is the chance that these links still accessible [5].

A. Publication Process of Scientific Data

Figure 1 illustrates the relation between data production and scientific publications. Several billion Dollars has been invested in the bottom level "primary data"[1]. Enriched with meta data, they would be a more efficient basis for life science resources as the "re-extraction" from articles [6].

In order to protect those investments and make the data reusable for later analysis, there is in general the agreement to support long-term primary data access. This argumentation is the basis for international efforts like the as ISO 14721 accepted reference model "Open Archival Information System" (OAIS) which aim is to preserve and share data for designated community. But, such comprehensive models are hard to realize for short or mid-term research projects.

One promising alternative is to motivate the publishing of data as research result into a "data paper" [7]. For this it is essential, to change the handling and acceptance of primary data within the scientific community. Data and publications have to become the same significant value for reputation [8]. The summarized benefits of the "data paper" are:

1) Scholarly credit to data publishers
2) Describe the data in a structured, human-readable form
3) Bring the existence of the data to the attention of the scholarly community

[1]The definition of "primary data" is not clearly fixed. For some it is a raw data stream from a device, also called "Level 0" data. For others, it is pre-processed raw data, but without scientific analytic processing steps. Still others do not differentiate the degree of processing, but consider all data, which is used for scientific publication as "primary data". In consequence, the subsequent elaborations consider all categories of primary data.

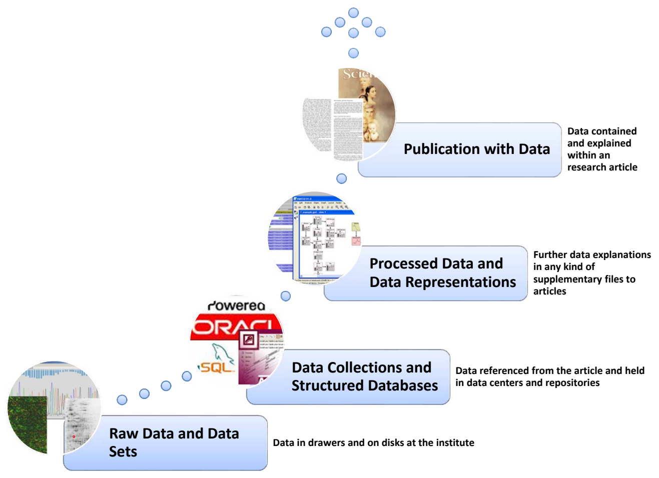

Publication with Data
Data contained and explained within an research article
Processed Data and Data Representations
Further data explanations in any kind of supplementary files to articles
Data Collections and Structured Databases
Data referenced from the article and held in data centers and repositories
Raw Data and Data Sets
Data in drawers and on disks at the institute

Figure 1. The data publication process (inspired by Gray et al. [4]) expresses the different manifestation forms of research data. At the *top layer* of the process, the journal, author, or scientist takes full responsibility for the publication including the aggregated data embedded in it and the way it is presented. For data published in the *second layer*, as supplementary files to articles, the link to the published "Record of Science" remains strong, but it is not always clear at what level the data is curated and preserved and if the criteria for discoverability and re-usability are met. At the *Data Collections and Structured Database layer*, the publication includes a citation and links to the data, but the data resides in and is the responsibility of a separate repository. At the *bottom layer* most datasets remain unpublished and consequently not accessible for later reanalysis.

In general many researchers are willing to share primary data with other scientists, but in fact only few of them really provide their data sets [9]. The access is often restricted to the project associated users. Even worse, the original data sets frequently remain in the hands of the producer or the researcher who processed the data. Another issue is the lack of meta data annotation. If journal or project policies do not force to use standard meta data, scientists tend to use personal meta data in form remarks or smart file naming.

An technical challenge is to prevent a loss of data. Long-term maintenance and backups of storage media should be available as service. In this context, loss of data also implies concepts, which enables to open and access files even in future. Approaches like "data format migration" converts an old data format lossless into a new one or "emulation" simulate the original software environment of the data format.

B. State of the Art

Domain specific *public primary data repositories* already exist. Examples are the NCBI "Sequence Read Archive" (SRA) for NGS raw data or the "Gene Expression Omnibus" (GEO) for gene expression data. The common concept is to use simple file transfer protocols, i.e. FTP or Aspera, for data upload using pre-defined file formats. In order to share this data, Web interfaces are the commonly provided method. Those infrastructures are commonly accepted as for data citation, but they are limited useable as general primary data storage. Such system can be cumbersome and inflexible when handling large NGS-files, i.e. several giga bytes of BAM or FASTQ. Such files are usualy container of several data items, whereas each item is represented as closed data unit, e.g. a sequence entry in a FASTQ file. Parsing a file to extract such items is a popular but inefficient method. In contrast, a random access to data blocks or items should be offered and implemented by suitable index structures. Such index would map an item to its absolute position in the file. In this regard, it shall be possible to efficiently retrieve a read set in chromosome 1 from position 100000-200000 or query a FASTQ sequence and its quality values using a given sequence name.

Beside public data repositories, there are a lot of systems that provide a *project level data infrastructure*. To share data popular systems are "Dropbox" , "Google Drive" and "network attached storages" (NAS). Here, the file exchange is frequently done using upload folder and shared user logins or even worse as email attachments. The consequences could be inconsistent or loss data, because of accidently file deletion, renaming or missing meta information to interpret the files. Exceptions are version tracking systems, e.g. "Subversion". Those are excellent to track the file history and prevent for accidently deleted files. Drawbacks in all the above systems are the limited support for meta data annotation and information retrieval. The common argument for their popularity is the seamless usability in the desktop environment. More dedicated solutions are project collaboration systems like "LabKey" [10] or Laboratory Information Systems (LIMS) [11]. But, its focus as collaboration platform and its highly integrated design makes it difficult to use as stand-alone and citable storage infrastructure.

The *annotation of primary data* with meta information is one important factor for its interpretation and structuring. This is reflected in manifold meta data schema that is used in life sciences. They are classified into technical and semantic ones. For instance, in system biology a review summarize 30 schemata and data exchange formats [12]. Apart from that, technical and administrative meta data cover aspects of management and processing of digital scientific resources. One popular and as ISO standard 15836 accepted schema is the "Dublin Core Metadata Element Set" (DCMES)[2] and the close related "DataCite Metadata Schema"[3]. DCMES was developed by scientists and librarians to homogeneously describe digital objects using 15 mandatory elements, like "`CREATOR`", "`FORMAT`" or "`DESCRIPTION`". The DataCite schema is less strict and comprise of 5 mandatory and 12 optional elements. But the most popular way of primary data annotation is still semantic enriched file names.

Information retrieval, using search engine technology is an efficient method to find relevant data. Tools like "Google

[2]Dublin Core Metadata Set: http://dublincore.org/documents/dces/
[3]DataCite Metadata Schema: http://schema.datacite.org/

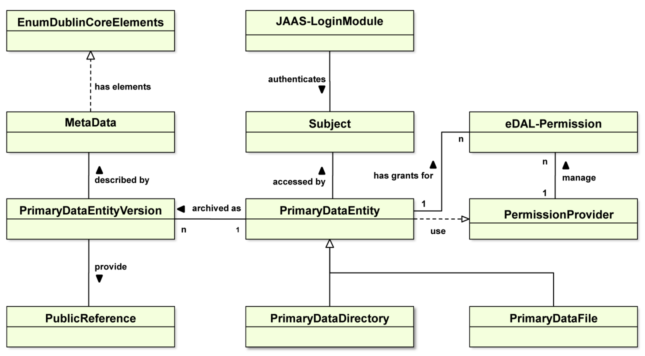


Figure 2. Overview about the concept and the most important parts of the *e!DAL* infrastructure

Desktop" are popular at the scientist desktop. The increasing availability and performance of this technology support the trend to replace classic query forms and boolean query languages by keyword based search and relevance filtering. This gets increased importance in life science information systems and is also implemented in primary data repositories, e.g. the "DataCite Metadata Search".

Since URLs are short-lived in practice, they are not as reliable references considered. Beside archive specific identifier, e.g. Genbank accession number, *persistent identifiers* are used in life sciences. Established identifier are the "Digital Object Identifier" (DOI), e.g "doi:10.5447/IPK/2012/0" or the "Life Sciences Identifier" (LSID) [13], e.g. "urn:lsid:example.org:namespace:ID" close this gap. Latter are the prerequisite for data citation in the sense of this manuscript.

II. Design and Concept of *e!DAL*

The mentioned challenges in primary data management towards a consistent data publication process lead to the conclusion for the need of a universal platform for primary data management. Therefore we summarized six general requirements for primary data storage. *e!DAL* stands for (***e**lectronical **D**ata **A**rchive **L**ibrary*). The UML class diagram in Figure 2 show the core elements of *e!DAL*. It provides a file system like storage system, but with enhanced features for long-term data preservation. In this section we introduce the concept of *e!DAL* and the implementation of the six elementary features.

1) Version Management: For long-term archiving primary data an infrastructure should provide a version control system to track the history of files and their meta data. The central classes of the data structure of *e!DAL* are the abstract `PrimaryDataEntity` class and the extended classes `PrimaryDataDirectory` and `PrimaryDataFile`, which make it possible to create directories and add files or subdirectories. Furthermore, *e!DAL* provide the `PrimaryDataEntityVersion` class to store multiple versions of a digital object with different meta data sets. The user can store a complete progress of an object and switch between different versions.

2) Meta Data: To make digital objects in the future reusable a standardized meta data schema is necessary to describe objects with information about the data. The `MetaData` class of *e!DAL* supports the *Dublin Core* standard to annotate every version with meta data. The schema did not specify which data type makes sense to describe each element, but *e!DAL* provides different simple data types to make the maintenance of the meta data user friendly.

3) Information Retrieval: For the acceptance of a data infrastructure it is important to store objects easily, but furthermore it is indispensable to provide efficient methods for the information retrieval. *e!DAL* provides functions to browse through the infrastructure, e.g. to list all objects in a directory or go backwards to the parent directory. With increasing number of objects this way is only comfortable, if you know the exactly path where the objects are stored. Therefore, it is possible to use a keyword search over selected meta data elements or in the entire meta data set. These queries may be executed over the entire data content or dedicated in specific subdirectories, which may be used to model project or experiment structures.

4) Persistent Identifiers: To ensure, that the digital objects are still citable, even if their URLs becomes obsolete, persistent identifiers are necessary. *e!DAL* provides the `PublicReference` interface, which allows to connect the infrastructure with different identifier systems to register a global persistent identifier for a digital object.

5) Data Security: The public access is important for the scientific community, but of course it is also necessary to protect the data against misusage. We integrated a fine grained, and efficient security system into *e!DAL*, which allows to authenticate and authorize different users or groups to use the methods of the infrastructure. To support different procedures of authentication or alternatively implement own login modules, *e!DAL* delegate the authentication process. Doing so, the JAVA embedded "JAVA Authentiction and Authorization Service" (JAAS) is used. All public methods and need a security check if the user is allowed to run them. With an aspect oriented programming by "AspectJ" [14] we implement a central security aspect, that weaves an authorization check around public methods. Otherwise the security check has to be implemented separately before every method. With "AspectJ" the source code stay clearly.

6) Easy Embeddable: The infrastructure has to be reusable for different application scenarios. Therefore *e!DAL* is designed to be usable in existing code of data frontends. Thus, it is designed similar to classical file system APIs. Main concepts like, hierarchical iteration over directory structure and the storage of binary data are available as known from the JAVA API. Furthermore, most code to access the storage backend is abstract. The `ImplementationProvider` is the central point that links all neccesary implementations, like authentication by a `Subject` object, the actual storage of binary data stream, information retrieval, meta data storage etc. In consequence, the application code need to choose a suitable implementation[4].

III. Results and Discussion

The presented API is a flexible and efficient add-on for JAVA based data frontends to enrich them with features to manage, cite and share primary data. The modular concept of *e!DAL* is designed to meet different application scenarios. In this section we want to give a compact overview of the multiple possibilities to use the *e!DAL* API.

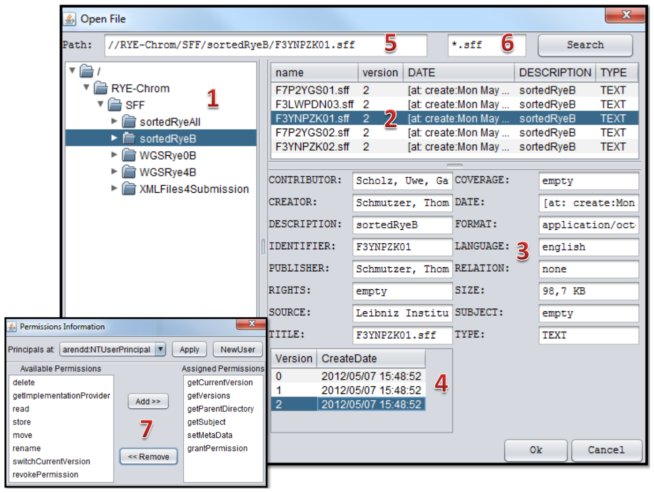
A. *The API Modules & Show Case*

To use *e!DAL*, the JAR archives have to be linked to the application code[5]. To do so one may use Apache Maven[6] to resolve all necessary dependencies or alternatively download the module JAR archives, which pack *e!DAL* and all dependent 3rd party libraries.

1) Core JAVA API: For the usage in stand-alone applications only the core API package is necessary. *e!DAL* works as a local storage system. The user can save and read his files, annotate them with meta data, save different versions, publish links and search for specific objects. *e!DAL* can support scientists by archiving their local primary data to guarantee a long-term usability. Normally with a simple file system you can only store meta data in separate files or sort all objects by a folder hierarchy.

2) Client-Server Interfaces: *e!DAL* uses the JAVA RMI technology and can operate as a central repository. Therefore it is possible that, e.g. scientists who working on the same project can share and parallel access their files. The security system can protect objects, e.g. to allow only the project partners to access them. It is not necessary to transfer the files to a separate server on which all partners can access or to use portable storage media to exchange files. When starting a new *e!DAL* server the user can specify a lot of parameter, like the mount path for the system or the port of the RMI registry. After starting several clients can connect with the *e!DAL* server and use all functions of the core API.

3) The FileChooser Graphical User Interface: Beside the usage as programming interface, we provide a graphical user interface (GUI), which can be embedded into an existing GUI and consists of a file open and save dialog. Figure 3 shows a screenshot of the *e!DALFileChooser* dialog as extension to the standard JAVA *JFileChooser*. It works like a file explorer to browse through the stored objects. It allows to



name	version	DATE	DESCRIPTION	TYPE
F7P2YGS01.sff	2	[at: create:Mon May ...	sortedRyeB	TEXT
F3LWPDN03.sff	2	[at: create:Mon May ...	sortedRyeB	TEXT
F3YNPZK01.sff	2	[at: create:Mon May ...	sortedRyeB	TEXT
F7P2YGS02.sff	2	[at: create:Mon May ...	sortedRyeB	TEXT
F3YNPZK02.sff	2	[at: create:Mon May ...	sortedRyeB	TEXT

CONTRIBUTOR:	Scholz, Uwe, Ga	COVERAGE:	empty
CREATOR:	Schmutzer, Thom	DATE:	[at: create:Mon
DESCRIPTION:	sortedRyeB	FORMAT:	application/oct
IDENTIFIER:	F3YNPZK01	LANGUAGE:	english
PUBLISHER:	Schmutzer, Thom	RELATION:	none
RIGHTS:	empty	SIZE:	98,7 KB
SOURCE:	Leibniz Institu	SUBJECT:	empty
TITLE:	F3YNPZK01.sff	TYPE:	TEXT

Version	CreateDate
0	2012/05/07 15:48:52
1	2012/05/07 15:48:52
2	2012/05/07 15:48:52

Available Permissions	Assigned Permissions
delete	getCurrentVersion
getImplementationProvider	getVersions
read	getParentDirectory
store	getSubject
move	setMetaData
rename	grantPermission
switchCurrentVersion	
revokePermission	

Figure 3. eDAL-FileChooser dialog. (1) - File tree to navigate through the stored directories. (2) - Window to display all files and subdirectories in the choosen folder. (3) - Textfields display the meta information of the choosen version. To change the meta information double click on a textfield. (4) - Table show all stored versions of an digital object. Switch between the version by mark a field. (5) Textfield show the complete path of the current object. (6) Textfield for search function. Enter a keyword and click on Search. (7) Right Click on an object to open the permission dialog to change the rights for an object

edit the meta data, manage the permissions and find specific objects with the search function.

4) WebDAV Server: In order to support non-JAVA clients and URL based access to *e!DAL* resources, a WebDAV module has been implemented[7]. It supports any HTTP browser to get files or WebDAV clients to mount an *e!DAL* server as remote file system. Thus, we are able to support classic file browser like “Windows Explorer”, Linux “KDE Dolphin” or any other WebDAV compatible platform. More technical details are provided in the documentation to the software demo in section III-B.

To show file versions and meta data, managed in *e!DAL*, artificial folders and files are displayed in WebDav directory structure: Each *e!DAL* file is (you see e.g. in the FileChooser GUI) comprise meta data and versions. A file is shown in WebDAV as folder including all versions of the *e!DAL* file and for each version a XML file, which makes it possible to show and edit meta data. It is also possible to save different versions and manage the meta data.

B. e!DAL *in Action - A Show Case*

In order to show the features of *e!DAL*, a demo has been implemented. It demonstrates how to start-up a primary data repository from scratch and is available as zip archive at the project website. As use case, the demo will import a snapshot of NGS data, start a panel of the *e!DALFileChooser* GUI, and initiate the WebDAV server at local machine (at default port 8080). The main component

[4]for this paper we provided an implementation called `FileSystemImplementationProvider`. It use H2, for meta data storage, Apache Lucene for information retrieval and file system for binary data storage.

[5]http://projects.ipk-gatersleben.de/eDAL-Project/download.html

[6]Apache Maven: http://maven.apache.org/

[7]Featured by the *McEvoy et al.: Milton-API*, http://milton.io/

is the *e!DALFileChooser* dialog, which shows core API functions. In parallel the WebDAV server is started. In order to present access to *e!DAL* a WebDAV compatible browser is required.

IV. Discussion and Future Work

A. Discussion

1) Implementation and Scalability: Beside the popularity of JAVA and the availability of skilled software developer, the major arguments to use this platform are the availability of industry standard frameworks like authentication services (JAAS), connectors to backend databases (Hibernate), code weaving (AspectJ) and build systems (Maven). And last but not least its widely acceptance as most popular programming language and its platform independent execution environment.

A number of data frontends in bioinformatics are developed using JAVA technology and can be seamless integrated into *e!DAL* infrastructure. In order to use *e!DAL* as storage backend in a non-JAVA environment or in non-interactive, scripted bioinformatics workflows, a WebDAV module is provided. It may be mounted as a standard file system. By mapping the versions and meta data *e!DAL* as virtual folder or XML-files, enhanced features are visible to any kind of software. Information retrieval interfaces are not supported by WebDAV or any other remote file system. In consequence, the API search features can be used only by JAVA *e!DAL* clients.

2) Backend Storage and Code Quality: The current backend implementation of the *e!DAL* infrastructure based on a local file system, i.e. FAT32 or EXT3, to store the actual data and a relational database to manage the meta data. Using timestamps for the files store and the ACID properties of the used H2 database system enables the use of *e!DAL* in a concurrent user environment.

Following own lessons learned in past decade of active software development in research projects, from published studies [15], guide lines for Agile software development and consequent automated testing (using JUnit) we were able to focused to develop the core code. On the one hand this guarant the use of up-to-date, best performing frameworks and, on the other hand, prevents from wasting time in developing proprietary, error-prone code for security, database connection, user management etc.

B. Future Work

Besides the completion of the WebDAV implementation, we work on the integration of the DataCite registration interface. The next version will implement methods to request a DOI from a registration agency and assign it to a stored object. In order to handle very large files, an extension for the information retrieval interface is planned. Furthermore, we work on a non-JAVA client application, in form of a REST web service interface.

Availability

All of the libraries, the maven artifacts, full API documentation, and samples are available on the *e!DAL*-website: http://projects.ipk-gatersleben.de/eDAL-Project/

Author's contributions

ML initiated the project. DA, ML, SF, and CC designed the data structure and tested the software. DA and ML implemented the core API. DA implemented the server and client API. JC implemented the eDALFileChooser GUI. ML and DA wrote the manuscript. US supplied use cases, NGS test data sets and supervised the project. All authors read and approved the final manuscript.

Acknowledgment

This work was supported by the European Commission within its 7th Framework Program, under the thematic area "Infrastructures", contract number 283496, and by the Leibniz Association in the framework "Pakt für Forschung".

References

[1] D. S. Roos, "COMPUTATIONAL BIOLOGY: Bioinformatics–Trying to Swim in a Sea of Data," *Science*, vol. 291, no. 5507, pp. 1260–1261, 2001.

[2] Z. Lu, "PubMed and beyond: a survey of web tools for searching biomedical literature," *Database*, vol. 2011, 2011.

[3] M. Y. Galperin and X. M. Fernndez-Surez, "The 2012 Nucleic Acids Research Database Issue and the online Molecular Biology Database Collection," *Nucleic Acids Research*, vol. 40, no. D1, pp. D1–D8, 2012.

[4] J. Gray, "Jim Gray on eScience: A Transformed Scientific Method," retrieved from http://research.microsoft.com/en-us/collaboration/fourthparadigm/4th_paradigm_book_jim_gray_transcript.pdf.

[5] N. R. Anderson, P. Tarczy-Hornoch, and R. E. Bumgarner, "On the persistence of supplementary resources in biomedical publications," *BMC Bioinformatics*, vol. 7, p. 260, 2006.

[6] D. Jameson, K. Garwood, C. Garwood, T. Booth, P. Alper, S. Oliver, and N. Paton, "Data capture in bioinformatics: requirements and experiences with pedro," *BMC Bioinformatics*, vol. 9, no. 1, p. 183, 2008.

[7] V. Chavan and L. Penev, "The data paper: a mechanism to incentivize data publishing in biodiversity science," *BMC Bioinformatics*, vol. 12, no. Suppl 15, p. S2, 2011.

[8] B. Nelson, "Empty archives," *Nature*, vol. 461, no. 7261, pp. 160–163, September 2009.

[9] V. S. Smith, "Data publication: towards a database of everything," *BMC Research Notes*, vol. 2, p. 113, June 2009.

[10] E. Nelson, B. Piehler, J. Eckels, A. Rauch, M. Bellew, P. Hussey, S. Ramsay, C. Nathe, K. Lum, K. Krouse, D. Stearns, B. Connolly, T. Skillman, and M. Igra, "Labkey server: An open source platform for scientific data integration, analysis and collaboration," *BMC Bioinformatics*, vol. 12, no. 1, p. 71, 2011.

[11] C. Stephan, M. Kohl, M. Turewicz, K. Podwojski, H. E. Meyer, and M. Eisenacher, "Using laboratory information management systems as central part of a proteomics data workflow," *PROTEOMICS*, vol. 10, no. 6, 2010.

[12] A. Brazma, M. Krestyaninova, and U. Sarkans, "Standards for systems biology," *Nature Review Genetics*, vol. 7, pp. 593–605, 2006.

[13] M. S. L. T. Clark, Tim, "Globally distributed object identification for biological knowledgebases," *Briefings in Bioinformatics*, vol. 5.1, pp. 59–70, January 2004.

[14] G. Kiczales, E. Hilsdale, J. Hugunin, M. Kersten, J. Palm, and W. G. Griswold, "An overview of aspectj," in *Proceedings of the 15th European Conference on Object-Oriented Programming*. Springer-Verlag, London, UK, 2001.

[15] K. Rother, W. Potrzebowski, T. Puton, M. Rother, E. Wywial, and J. M. Bujnicki, "A toolbox for developing bioinformatics software," *Briefings in Bioinformatics*, vol. 13, no. 2, pp. 244–257, 2012.