Information Services & Use 37 (2017) 335–342
DOI 10.3233/ISU-170852
IOS Press

Data-first manifesto: Shifting priorities in scholarly communications

Clifford B. Anderson
Associate University Librarian for Research and Learning, Vanderbilt University, 131 21st Ave S, Nashville, TN 37203, USA
Tel.: 615-322-6938; E-mail: clifford.anderson@vanderbilt.edu

Abstract. This paper introduces and exegetes the Data-First Manifesto, which calls for prioritizing data curation over interface design in digital scholarship projects as well as for rethinking how to foster scholarly communication in the performance of digital scholarship. The origins of the manifesto are discussed and its four sets of ordered preferences are detailed together with a discussion of how the implementation of these shifts in priorities might transform scholarly communications in the field of digital scholarship.[1]

Keywords: Digital scholarship, data curation, digital-first manifesto, data citations, data serialization

1. Origins of the data-first manifesto

The Data-First Manifesto emerged from growing frustration that digital scholarship projects were getting hung up or stalled for technical reasons that had nothing intrinsically to do with their scholarly aims. As Associate University Librarian for Research and Learning at Vanderbilt University, I consult regularly with faculty about digital projects. Those conversations frequently turn on questions of technical implementation. How can I put my project on the web? Should I use Omeka or Drupal or some other content management system to host my data? Who can customize the web design for my project? How can we wire together these two components? How can I requisition a server, stable IP address, and domain name to deploy my project? There is a critical shortage of technical staff at Vanderbilt to address these questions. Projects flounder as scholars search for technical resources to achieve their visions.

Digital scholarship falls into a gap between scholarly production and information technology. Most university Information Technology (IT) departments do not have the resources to support bespoke digital projects. Their focus is on matters such as maintaining the network environment, supporting infrastructural components like Enterprise Resource Management (ERM) systems, and desktop support. Some faculty turn to external vendors, but communicating requirements can be challenging; of course, contracts with external vendors will also typically be costly. In lots of cases, scholars working on digital projects find their way to librarians, who bridge the gap between research faculty and information technologists. While they can support some of these projects, there is always a practical limit to how much

[1]Previous versions of this paper were presented at the NFAIS 2016 Humanities Roundtable, *Digital Humanities – Preserving the Past, Capturing the Present & Building the Future* (September 23, 2016) and the NFAIS 59th Annual Conference – *The Big Pivot, Re-Engineering Scholarly Communication* (February 27, 2017).

librarians can do, especially if tasked with assignments other than digital project application development. If librarians do not articulate criteria for deciding which projects to support and at what level, faculty members may perceive them as making biased or unfair decisions. Why does so-and-so's project receive library support and not mine?

From a librarian's perspective, supporting scholarly projects raises questions of digital preservation. Over time, libraries may wind up hosting dozens of scholarly projects running on a wide variety of platforms. The components that make up these platforms – from programming languages to databases to web frameworks – require regular updates for security reasons. And, as anyone who has had to maintain software applications over a long period of time knows, making these updates frequently breaks things. While there are certain applications that require software preservation, most do not. What's necessary is to export the data, preserve it, and carry it forward into a new system. The effort that goes into patching old platforms might be redirected to data curation and the "extract, transform, and load" (or ETL) procedures needed to move data between platforms.

In face of these difficulties, why do so many digital scholarship projects begin by putting together interfaces with token datasets? At the Vanderbilt THATCamp in October 2015, Todd Hughes, former Director of Instructional Technologies at the Center for Second Language Studies at Vanderbilt, and I organized a session called the *Data-First Manifesto*.[2] The majority of attendees at the session were graduate students. We talked about how placing an emphasis on creating digital interfaces imposed barriers to their participation in digital scholarship. We also began to sketch an alternative approach.

The issues raised ranged from the hermeneutical, namely that interfaces place interpretative perspectives on multivalent data, to the equitable, with graduate students voicing concern that their contributions do not show through interfaces adequately. By the end of the session, we had the makings of a manifesto we wanted to write.

Todd Hughes and I worked back and forth during the spring of 2015 to compose a manifesto that we aimed to launch at the 2016 Digital Humanities Conference (DH2016) in Krakow, Poland. We borrowed its form from the Agile Manifesto, a call issued by leading software engineers in 2001 to shift priorities in project management.[3] Like the Agile Manifesto, we presented our argument in four sets of ordered priorities. The purpose of these theses is not to set out an entire theory of digital scholarship project management or even data curation. Rather, our goal is to shift the emphasis in whatever framework scholars are using to produce digital scholarship. As planned, we publicly presented the manifesto in Krakow. We received generally good feedback, but we're still in the listening phase as we gather critical responses to our suggestions.

2. Exegesis of the manifesto

The manifesto puts forward four ordered sets of preferences, calling for emphasizing data serializations over databases, application programming interfaces over graphic user interfaces, curating data over summarizing information, and citing datasets over referring to results.[4] I'd like now to exegete these four sets of preferences. How do these preferences promote the "agile, collaborative, efficient, and transparent practices in digital scholarship" the manifesto seeks to foster?

[2]See http://vanderbiltuniversity2015.thatcamp.org/.
[3]See [3].
[4]See [2].

3. Data serializations over databases

The first preference prioritizes "Data Serializations over Databases." Let's begin by defining our terms. By "serialization," I mean making data available in a format that can be preserved, read, and reused independently of the database in which those data are stored. Generally speaking, this means making data available as some form of structured text. Serialization often involves reading out data from a database into a format like CSV, JSON, RDF N-Triples, or XML, though there are other options. Whatever the format selected, the purpose of serializing data is to produce data that can be preserved, shared, and moved between applications.

What makes serialized data distinct from data in a database? After all, when displaying a table of data in a relational database through an interface, it looks like a CSV or Excel file with columns and rows. Actually, this view is itself represents a kind of serialization; databases store data internally in whatever manner makes querying and retrieval most efficient. Moreover, not all kinds of databases are equal when it comes to exporting data. As Dean Allemang and James Hendler remark, "There is no standard serialization language with which one can completely describe a relational database in such a way that it can be automatically imported into a competitor's system."[5] Generally speaking, you can produce CSV (or JSON or XML) from SQL queries. Most relational databases will allow you to export the entire database in the form of SQL expressions that you can use to reimport the data into the database, if needed; this serves the purpose of backing up the data, but does not necessarily make the data portable.

Other database models may be more capable of exporting data in standard formats. Given the ubiquity of relational databases, the tendency exists to equate databases with the relational model. The rise of so-called NoSQL databases during the past five or so years requires us to broaden our concept of databases.[6] The NoSQL moniker refers to a diverse class of databases, from document-oriented databases like CouchDB[7] to graph databases like Neo4j[8] to XML databases like BaseX,[9] and many more. Document-oriented databases like CouchDB straightforwardly serialize data to JSON since they represent data as JSON. The same point applies *mutatis mutandis* to XML databases like BaseX, which represent data as XML. Selecting a NoSQL database may reduce the distance (and dissonance) between data as stored in the database and data as serialized.

The main point of this ordered preference is to suggest making your data readily-available in a serialized format like CSV, JSON, RDF, XML, etc. Ideally, you would create scripts to generate a database from this serialized data. Of course, regenerating your database from files on your file system may not be possible if you're building a highly-interactive system that creates, updates, or deletes data in response to environmental changes. In that case, you might interpret this preference in the opposite manner, namely to script regular exports of the database to a serialized format.

4. Application programming interfaces over graphic user interfaces

The next preference puts "application programming interfaces" above "graphic user interfaces." As with the previous set of ordered preferences, scholars are more familiar with the concept of a graphic user

[5]See [1], p. 55.
[6]See [7].
[7]See http://couchdb.apache.org/.
[8]See https://neo4j.com/.
[9]See http://basex.org/.

interface (GUI) than an application programming interface (or API). Most digital humanities projects invest significant effort when creating GUIs. Indeed, many digital humanities project launch with little *but* an attractive GUI. These days, there are lots of nicely-designed open source templates available for websites, meaning that you can quickly announce a project on the web with little data, but a slick interface.[10]

Starting with an API, by contrast, place priority on communicating data. An API provides some means for software applications to access data from a project. There are several competing ways to provide API access, but a style called REpresentation State Transfer (or REST) has become the *de facto* standard.[11] In a REST-style API, you provide access to data through a set of carefully-designed URLs. For instance, you might access all the titles in a corpus via the URL /corpus/title. In a corpus of poetry, for instance, this URL would return a series of titles of poems along, perhaps, with other metadata about them. A call to /corpus/title/Rebelle would return any poem with the word “Rebelle” in the title, while a call to /corpus/id/FDM-000-000-000-095 would return the poem with that unique ID. Putting together a coherent API takes effort, especially since an API should allow you to combine requests for items, for example, by certain authors with certain words in the title. To be explicit, an API will not typically return HTML in response to these requests. Instead, the request will return data in a structured format like JSON, XML, or perhaps an RDF serialization like turtle.

Offering access to your application data through an API puts control in the hands of your users. Users may want, for instance, to combine information from your project with their own data or perhaps data from a third-party source to create a mashup. Ideally, a project would also develop its graphic user interface around its own API. That is, a project would not allow calls to be made directly to its data store from its GUI, but provide access only through its API. In this way, a GUI would exemplify the kind of rich interaction with the data made possible by its API rather than forming a barrier to the data in the project.

However, is designing an API is any easier than putting an interface online? After all, there are abundant tools for deploying spiffy interfaces. What out-of-the-box resources exist for designing APIs? While not as many tools exist, there are some emerging. The Swagger specification for designing APIs,[12] which major industry players, including Adobe, Google, and IBM, have adopted and re-dubbed the Open API Initiative,[13] standardizes the description of REST APIs across vendors.[14] According to its documentation, “The goal of The OpenAPI Specification is to define a standard, language-agnostic interface to REST APIs which allows both humans and computers to discover and understand the capabilities of the service without access to source code, documentation, or through network traffic inspection.”[15] There are open source and commercial options available for developing, deploying, and testing OpenAPI endpoints. Still, these specifications and tools target professional developers and may pose a barrier that is as insuperable to digital scholars as programming a custom interface.

As a lightweight alternative to (or perhaps a way station toward) implementing the OpenAPI specification, consider the approach taken by the University of Pennsylvania

[10]See, for instance, the Bootstrap framework (http://getbootstrap.com/).
[11]See [5].
[12]See http://swagger.io/specification/.
[13]See https://www.openapis.org/.
[14]See [14].
[15]See https://github.com/OAI/OpenAPI-Specification.

Library with OPenn.[16] OPenn provides a simple interface to its data, which consists primarily of images and TEI metadata. The three primary means of access are HTTP, FTP, and rsync. The HTTP interface is spare, resembling a file browser. If you click through the folders, you'll reach the metadata and images. This interface functions as a rough-and-ready API because it employs a consistent directory and file structure, allowing software developers to write code to retrieve data in bulk and also to make educated guesses about the URLs of resources.

Taking an API-first approach does not mean, of course, that you cannot also build a rich graphic user interface for the application. Ideally, you'd build your interface on the basis of your API. If you choose to work with the OpenAPI specification, for instance, you'd produce human-readable documentation for accessing your data (similarly to OPenn). You'd then design a graphic user interface with virtually any programming language (likely a JavaScript framework) to present your data in a rich, interactive format for your users.

The OPenn approach is beginning to shift the perceptions of editors of digital scholarly editions. In "Let's get nekkid! Stripping the user experience to the bare essentials," Daniel Paul O'Donnell and his co-presenters at the Digital Scholarly Editions as Interfaces conference at the University of Graz in September 2016 noted that OPenn inspired them to "reverse our assumptions" about their digital edition of Anglo-Saxon cultural heritage, titled "The Visionary Cross".[17] "A project like the Visionary Cross, we believe, is less about the interface than the quality of its data and the usefulness of its API."[18]

5. Curating data over summarizing information

The next thesis calls for "curating data over summarizing information." The leading idea here is that digital scholarship projects should, so far as possible, allow the data to speak for themselves. While providing a narrative (or interpretative visualizations) of data can provide a helpful snapshot of what the data contains, emphasis should be placed on making data self-describing and perspicuous.

The idea of self-describing data originates, at least in part, from the XML community, which aspired to make XML tags and attributes both human readable and machine interpretable.[19] XML is sometimes perceived as "verbose" because of this design commitment. Still, the Text Encoding Initiative (or TEI) provides a good model for creating self-describing data. Even those who have never worked with the TEI will be able to garner a general sense of the purpose of its elements and attributes from reading their names. The TEI website provides definitions and usage information about the elements and attributes.[20] On a more technical level, the TEI schema prescribes how its elements and attributes may be used to annotate data, ensuring that datasets conform to its (flexible) data model.

XML is less popular in the general programming community than it once was.[21] JSON has essentially taken over from XML among web developers who need to send packages of information across the web. While JSON certainly has its place given the ubiquity of JavaScript these days, switching from XML to JSON has not provided an advance in interpretability. The syntax is tenser and more compact, making it

[16] See http://openn.library.upenn.edu/; see also [11].
[17] See http://visionarycross.org/.
[18] See [8].
[19] See [12].
[20] See http://www.tei-c.org/Guidelines/P5/index.xml.
[21] See [9].

easier for machines to parse, but harder for humans to read.[22] Formats like JSON-LD, which use JSON to produce linked data, support something like the XML community's goal of making data both machine and human-readable.[23]

Among the biggest challenges come when sharing data as CSV files. It is still common to receive CSV files without metadata or supplementary descriptive documents. The guesswork that goes into interpreting these bare CSV files makes data analysis prone to errors and mistakes. It's easy, for instance, to mistake units of measure or to confuse data types. Fortunately, there are now good guidelines for describing tabular data. As Jeni Tennison has shown in her work toward "Linked CSV," it's possible to package metadata into CSVs through clever descriptive practices.[24] Recently, Tennison led a working group at the W3C with Gregg Kellogg and Ivan Herman that released several recommendations, including a "Model for Tabular Data and Metadata on the Web."[25] While the W3C working group wrapped up in March 2016, a "CSV on the Web" community group is carrying on the discussion under the W3C aegis.[26]

The goal here should be akin to the old creative writing adage of "show, not tell." A narrative or visualization *tells* users about the data, but ideally you would *show* how to produce your visualizations with your data. If you provide users with nicely-curated datasets, then they can not only reproduce and check the validity of your summaries, but also produce new analyses and visualizations to serve their particular aims.

At Vanderbilt, we've been conducting this work retroactively. In recent years, we've appointed a CLIR postdoctoral fellow for data curation, Veronica Ikeshoji-Orlati, to assist us with retrospective data curation projects. Recently, for instance, we collaborated on a project to curate the data in a digital humanities project. The Vanderbilt Library had hosted a digital edition of a book, *Charles Baudelaire: Une Micro-Histoire* by Raymond Poggenburg, that chronicles events in the life of the poet Charles Baudelaire (1821–1867).[27] The site is aging and we became concerned that the data behind the interface was becoming increasingly difficult to access. The data curation project transposed the data from a relational database into MODS XML documents. Ikeshoji-Orlati described all the data types in detail and also cleaned up the dataset. We are now preparing to release the dataset under a Creative Commons license for others to reuse. The hope is that scholars of Baudelaire will take the data and create interesting by-products that we did not originally envision, such as timelines, network analyses, and geospatial representations.[28]

6. Citing datasets over referring to results

The final preference prioritizes "citing datasets over referring to results." In a way, this preference is the mirror image of the previous one. Whereas the prior thesis addresses scholars who are developing digital scholarship projects, this clause speaks to scholars who engage with others' projects. When drawing on digital projects to warrant arguments, are scholars relying on the summaries of data that they find on the

[22]This point is, of course, debatable. Some apparently find JSON easier to read than XML. See [4]. It's probably a question of how familiar you are with the syntax of these formats.

[23]See http://json-ld.org/.

[24]See http://jenit.github.io/linked-csv/.

[25]See https://www.w3.org/TR/2015/REC-tabular-data-model-20151217/.

[26]See https://www.w3.org/community/csvw/.

[27]See [10].

[28]See [6].

website (or research articles) or are they engaging with the datasets themselves? Assuming that projects make datasets readily accessible, our preference is that scholars test their claims directly against the data rather than rely on others' claims.

Ideally, scholars would encourage others to cite their datasets by making them straightforward to access, explore, and cite. Making datasets accessible generally involves curating them, putting them in a data repository, and attaching permissive licenses which encourage reuse. Additionally, researchers should make it easy to explore how they reached their conclusion by providing their ETL (Extract, Transform and Load) procedures and software code for others to examine and reuse.[29] This second point touches, of course, on the concept of reproducible computational research.[30] Finally, researchers should make it easy to cite their datasets. While you can always request that downstream users cite your dataset (and any funding agency that supported its creation) according to a form you specify, a best practice is to associate a Document Object Identifier (DOI) with your dataset. Using a DOI allows you to document the subsequent uses of your dataset more accurately. DOIs also allow you, in most cases, to track changes to datasets, though there is ongoing discussion about the best way to handle DOIs for dataset versions.[31]

Steve Baskauf, senior lecturer in biological sciences at Vanderbilt, follows these principles in his *Bioimages* project.[32] If you visit the bioimages website, you probably won't come away impressed by the interface. The website offers a high-level orientation to the data and its licensing terms. The data are in a GitHub repository, which contains the CSV files and XQuery code that produce the site.[33] The GitHub repository also provides a list of the major releases of the dataset along with details about significant changes.[34] Finally, Baskauf has associated a DOI with every version of the dataset, making each independently citable. The date and DOI of the latest release is indicated on his website.[35] With its minimalist website and its citable datasets, *Biomages* incorporates the preferences of a data-first approach to digital scholarship.

7. Conclusion

Publishing a manifesto is, of course, presumptuous. Who are Todd Hughes and I to suggest new ways to carry out digital scholarship? And, admittedly, our manifesto hasn't taken over the world. At this point, there are a handful of signatories. Our hope in promoting these preferences was to help kick-start a conversation among digital scholars and, especially, graduate students and postdoctoral fellows, about how to shift the emphasis away from flashy websites and toward data curation, data publication, and data citation. In the end, it comes down to the economics of time. While knowing how to design an attractive interface is a great (and potentially lucrative) skill, scholars should not have to become web programmers to launch digital scholarship projects. If the manifesto succeeds in promoting this view, expect to see more stripped down websites offering richer access to scholarly datasets.

[29] See: https://en.wikipedia.org/wiki/Extract,_transform,_load.
[30] See [13].
[31] See https://github.com/ResearchSoftwareInstitute/software-data-citation-ws/issues/19.
[32] See http://bioimages.vanderbilt.edu/.
[33] See https://github.com/baskaufs/Bioimages.
[34] See https://github.com/baskaufs/Bioimages/releases.
[35] See https://zenodo.org/record/51121#.WLIqKRIrLBL.

About the author

Clifford B. Anderson is the Associate University Librarian for Research and Learning at Vanderbilt University. He holds a Ph.D. in Theology from Princeton Theological Seminary and a M.Div. from Harvard Divinity School as well as an M.S. in L.I.S. from the Pratt Institute in New York City. Among other publications, Anderson is co-author of the forthcoming *XQuery for Digital Humanists* (Texas A&M University Press).

References

[1] D. Allemang and J. Hendler, *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL*, Elsevier, 2011.

[2] C. Anderson and T. Hughes, Data-first manifesto, 2016. http://datamanifesto.info/.

[3] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning et al., Manifesto for Agile software development, 2001. http://www.agilemanifesto.org/.

[4] D. Crockford, JSON: The fat-free alternative to XML, 2006. http://www.json.org/xml.html.

[5] R. Fielding, Architectural styles and the design of network-based software architectures, PhD thesis, University of California, Irvine, 2000. http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.

[6] V. Ikeshoji-Orlati and C. Anderson, Developing data curation protocols for digital projects at Vanderbilt: Une Microhistoire, in: *12th International Digital Curation Conference*, Digital Curation Centre, Edinburgh, 2017.

[7] D. McCreary and A. Kelly, *Making Sense of NoSQL: A Guide for Managers and the Rest of Us*, Manning Publications, Shelter Island, 2013.

[8] D.P. O'Donnell, Let's get nekkid! Stripping the user experience to the bare essentials, in: *Digital Scholarly Editions as Interfaces*, Graz University, 2016. https://static.uni-graz.at/fileadmin/gewi-zentren/Informationsmodellierung/PDF/dse-interfaces_BoA.pdf#page=15.

[9] A. Patrizio, XML is toast, long live JSON, *CIO*, 2016. http://www.cio.com/article/3082084/web-development/xml-is-toast-long-live-json.html.

[10] R.P. Poggenburg and Jean and Alexander Heard Library, *Charles Baudelaire: Une Micro-Histoire*, 2nd expanded edn, Vanderbilt University Press, Nashville, Tenn., 2001. http://www.library.vanderbilt.edu/poggenburg.

[11] D. Porter, What is an edition anyway? A critical examination of digital editions since 2002, in: *Digital Scholarly Editions as Interfaces*, Graz University, 2016. (PowerPoint) https://static.uni-graz.at/fileadmin/gewi-zentren/Informationsmodellierung/PDF/Interfaces_program_final.pdf.

[12] E.T. Ray, *Learning XML: Creating Self-Describing Data*, O'Reilly Media, 2003.

[13] Sandve, G. Kjetil, A. Nekrutenko, J. Taylor and E. Hovig, Ten simple rules for reproducible computational research, *PLoS Comput Biol* **9**(10) (2013), e1003285. doi:10.1371/journal.pcbi.1003285.

[14] T. Wemett, SmartBear launches open API initiative with key industry leaders including Google, IBM and Microsoft, *Business Wire*, 2015. http://search.proquest.com.proxy.library.vanderbilt.edu/docview/1729802783/abstract/D4B28449CB344A82PQ/1.