Any changes in the record must get saved (which is the final step for each record).

[side comment: OLE uses | as the delimiter as opposed to $, $$, or the ‡ of Connexion.]

At this point, the bulk of the headings from this vendor are getting $0s and $1s—and only a small fraction need this manual step (which is good)—and our data, even from vendors, is better prepared for a Linked Data future (especially as we migrate to a schema-agnostic FOLIO that can theoretically move to BIBFRAME in the future too).

So, herein I detail a nitty, gritty headings cleanup process [we are almost done, whew!] that involves a library-wide process and some manual steps that happen with records loaded for the law library. Also, I wanted to talk a little more about Kuali OLE since it gets almost nothing written about it and is about to go away. I hope to follow up with at least one more Description & Entry column exhibiting Kuali OLE before it is closed down.

THE INTERNET

Navigating GitHub for meaningful communication about software platforms (with no intention of learning git or coding) – Part 1

Wilhelmina Randtke
Florida Academic Library Services Cooperative

What is git?

Git is a programming language specifically for versioning code or any other text file. Rather than recording only incremental saves, like a version history on a document might do, git uses small changes (commits) and allows multiple people to work together by saving their own changes, sharing out changes for possible inclusion in a release, locating and incorporating other people's changes, and commenting on each change. Git is fairly complicated to learn and conceptually involves there being multiple realities for any piece of software, rather than a single version which changes over time.

Git also involves leaving a comment on each saved commit to a file or set of files to explain why the change was made. These comments can be retrieved later when looking at a specific piece of code.

Git is generally used for software development, communicating between software developers about software changes and versions of in-development software, and keeping notes on history, such as why a change was made.

Generally, releases of software are not put out to the public through git, but rather are done through sets of static text files (i.e. a zip or tarball to download and install). Git is for a deep dive into a single platform or the history of a piece of software. Most people working with a software platform just want to use it and not take a deep dive into the inner workings. Each deep dive takes time, and for end users and organizations running many software programs and platforms, there isn't time for a deep dive into everything. For this reason, most releases of software that you would download and run to power a website are released as .zip files without git.

Instead, git is for collaboration, new development, and deep dives into a single software platform.

Think of git as, at heart, being about communication. A culture and series of formalities for talking to other people. Not a language for talking to computers.

What is GitHub?

Various websites exist for hosting code versioned in git. The two most significant in the U.S. and Canada are GitHub (https://github.com) and Bit Bucket (https://bitbucket.org). (Bit Bucket is run by Atlassian Corporation, so may sometimes be referred to as Atlassian; Atlassian also runs several other software collaboration platforms, including Jira and Confluence.)

Each website for hosting files versioned in git has a graphic interface, as opposed to git which is all command line. The graphic interface is important because it allows communication with a wider group of people, specifically communication back and forth with people who don't know git and with people who are uncomfortable with a command line interface. The graphic interfaces aren't simple and, on some level, involve understanding and navigating multiple realities for any piece of software, but they are less intimidating and don't require memorizing and typing commands. So, the conceptual complexity of the communication is still a factor, but learning a new programming language is not.

Regarding GitHub and Bit Bucket, GitHub is geared toward public access, whereas Bit Bucket has mostly private activity. Both GitHub and Bit Bucket offer paid accounts as well as free-of-charge accounts, and both have a free-of-charge tier allowing private repositories which can be shared with a specific select group of other users. Both also allow publicly available repositories through the free and paid accounts. See https://github.com/pricing and https://bitbucket.org/product/pricing?tab=self-managed. The difference is not in what each platform allows, but rather in how each site tends to be used by people and companies. The major use of Bit Bucket is to allow collaboration on a software project without making the code public. For example, a large corporation might allow all employees access to the company Bit Bucket accounts. Code can be made public on Bit Bucket, but it is not the norm, and most activity on the site is private. Meanwhile, GitHub has a huge amount of public activity. GitHub is commonly used by people and companies to self-promote and raise the profile for their organization and projects by posting code and contributing code to other projects, as well as for projects to gather input and contributions from a wide community in a controlled way.

In short, GitHub is commonly used as a public site for sharing, self-promotion, and collaboration. The public aspect is why it's well known, including awareness of the site in a wider community than just technical. Anyone can search GitHub and grab files and info from it, all without needing a login or account. And there's lots of content to search, navigate, and pull from.

Also key is that not everything that's open source is on GitHub. GitHub is just really popular, and many projects have a public face there in order to be visible. But absence of activity on GitHub doesn't mean an absence of activity.

Basic background: A software platform can be spread across many repositories.

In GitHub, each user account can own multiple repositories. The user account is something you are likely familiar with from all kinds of websites—a user account's public activity appears at https://github.com/useraccountnamehere. Within each user account, there can be multiple repositories. Each repository is a versioned file or set of files. Generally, that's source code for software, but the content of a repository can be any text file.

When you go to look at a software platform, you are not necessarily (almost never!) looking for one single git repository where code is maintained. Instead, components of a larger software platform might be maintained separately. Also, as discussed earlier, supporting libraries, which are necessary to run a platform but are used by many platforms in an out-of-the-box fashion, are typically bundled into a .zip or tarball release but are not published on GitHub as part of the GitHub public face of a software platform using that library. That would be clutter.

(An example of a supporting library might be software to automatically make and save thumbnails from bigger pictures. Another example of a supporting library might be code for allowing Chinese, Arabic, and Russian/Cyrillic characters to be stored and displayed. Another example of a supporting library might be code for displaying letters as an image for captchas. These can be used in an out-of-the-box way and do not need to be customized or written from scratch for projects across the web.)

Then, a single platform and the parts of it which are unique are frequently spread across multiple repositories. When you go to look at a software platform, you often are looking at one central user account in GitHub, then at several different repositories, each tracking pieces of the code. Once again, it's a deep dive to understand how those repositories interact with one another. If you are looking to connect to open source without learning git and without taking a deep dive, then the important thing is to understand that one software platform might live in multiple repositories; that's a common and normal thing to see, so be comfortable with it. Be comfortable with there being multiple small parts that come together to make a whole and those parts not necessarily being laid out together.

In trying to locate the public face of the code for a software platform on GitHub, you are usually looking for one authoritative user account; then, within that single user account, you are often looking at multiple repositories. And most of the actual information you need is found at the repository level.

LIBRARY METRICS

Library Metrics Trends: What's On The Horizon?

Rachel Decker
Hugh & Hazel Darling Law Library
Chapman University Fowler School of Law

In the last *Library Metrics* column, I wrote about the U.S. News & World Report's new Scholarly Impact Ranking. Impact metrics are not necessarily new, but perhaps signal a trend that is worth paying attention to. As we come to the end of 2019, I want to look ahead and provide a brief trend analysis in library metrics. An article that appeared last year in *The Serials Librarian* gave me a good starting point (Markovic and Oberg 2018), and I intend to build on some of