****

**Documenting the DH Process**

**Notes for Week 12: Module 12**

\*This post includes direct quotes from Jessica Dussault (University of Nebraska-Lincoln)\*

Video: The Future of Your Data by Jessica Dussault

Now, let’s learn about why DH projects must all come to an end and how to plan so that you can keep them running as long as possible.— “all things turn to dust”

**Introduction**

* Technology changes
* The people who created projects and know how they work have moved on or retired
* Planning for longterm of project
  + Storing things
  + Image what your project would look like in one year or five
* Topics covering:
  + 1. Why nothing lasts forever
  + 2. How to design your project to help it gracefully age

**Why Nothing Lasts Forever**

* Threats that you face when creating digital humanities:
* Physical (hardware) limitations
  + Losing a thumbdrive
  + Dropping phone
  + Component failure– because it ages (the silver components are tarnishing)
  + Bit rot –file may be corrupted in a few years
  + Probably more things!
* Physical mitigations:
  + Backup your stuff!
  + Organize, migrate files every few years
  + Checksums– when calculate checksum, it should be the same–if does not match, then it probably changed
  + Controlled storage (cold, low humidity)
* Software Limitations
  + Can’t open a file format–an old file
  + Unable to run a script–and it doesn’t run anymore
  + Website broken
* Software Mitigations
  + Complete redo in new technology
  + Upgrade!
  + Combination of the above
  + What was the original like?
  + Do you have any ways of getting it running (emulator, etc)?
  + What were the important parts, and can you salvage them? –the more you can pull out
* Provider Limitations (provider-third party–like wordpress or Flickr or cloud services)
  + Unexpected expenses–like icloud
  + Company goes out of business
  + Changes to laws/regualtions/policies
  + Malicious actors, (DDOS, etc)
  + Retiring products or services
* Provider Mitigations
  + Make a plan for the worst case
  + Be aware of emails, correspondence from provider
  + Keep a copy of your data
* Human Limitations
  + The original developer is gone
  + No instructions
  + Not enough knowledge to fix or redo
* Human Mitigations
  + Write everything down and put in a place you can find it
  + Choose commonly used tech
  + Make a plan for the future

**Planning Your Digital Project’s Life**

* Before you Get Too Far
  + What’s the “minimum viable” product?
  + Choose or create standards
    - Metadata (not too much shit though!)
    - File naming conventions
    - Where you put stuff
    - Scanning resolution, photo quality, etc.
* Collaborate!
  + Identify collaborators (organizations, people)
  + Who is responsible for what?
  + How will you communicate?
  + How do you credit everyone?
* Set up Documentation and Issue Tracking
  + Find location for documentation
    - Leave documentation with code–although this may not always be feasible
  + Think about technology for documentation!
  + How will you keep track of TODO items?
  + Who will be able to access all of the above?
* Choose Tech Carefully
  + What do you need it to do?
  + Is it widely used and supported?
  + How much maintenance will it need?
  + Can you easily train someone to use it?
  + Will your data be stored within the system or outside of it? — word, docs, mp3–proprietary format
  + Plain text and non-proprietary formats?
* Think ABout Money
  + Figure expenses now AND in the future
    - Website & file hosting
    - staff/consultant wages
    - Software purchases
* Make Plans
  + Will the project ever “end”?
    - How will you add new things?
    - What if you find mistakes?
  + What if you are asked to take part down?
  + What if part of the site stops working?
  + How might you gracefully retire it?
* What’s Next?
  + Think about the decline of your digital projects and how to keep them going as long as possible

**The reading:** <http://www.digitalhumanities.org/dhq/vol/13/1/000411/000411.html>

* Legacy Portfolios
* Policy context
* Software development life-cycle (SDLC) and Infrastructure
* Principles
* Implementation
* Conclusion
  + “King’s Digital Lab has implemented pragmatic processes that take into account the human, as well as the technical, financial and political perspectives implicit in digital scholarship. It has reinforced the lab’s commitment to producing digital research within a holistic and scalable framework, supported by straightforward documentation to ensure mutual clarity about what can be expected from research partnerships. A key component of this framework includes the enhanced Software Development Lifecycle (SDLC) process, which is now implemented from the inception of a project, to align its development with post-publication maintenance and, where appropriate, archiving plans. Early clarity about the feasibility and cost of maintaining projects beyond the funded period allows all parties time to plan ahead, with sufficient time to accommodate the development and turnaround time of follow-on funding applications, negotiations with partner institutions, infrastructure resourcing and requisite allocation of staff time. In addition to optimising maintenance and management of legacy digital research outputs, this approach minimises ambiguity regarding responsibilities and expectations, and contributes to reputation risk management in more than one dimension. Contrary to what might have been expected, KDL’s experience of introducing the level of transparency and process described in this article was almost uniformly positive.”

website/game: <https://archive.org/details/software?and%5B%5D=mediatype%3A%22software%22>

**TUTORIAL NOTES** (Tutorial video by Jessica Dussault)

**Voyant**

<https://voyant-tools.org/>

More info about it: <https://voyant-tools.org/docs/#!/guide/tools>

You can either paste in text, upload documents, or give it a URL. Keep in mind that URLs, may also include things like the header / navigation / footer portion of the website and skew your results. Cleaning up text before doing any analysis on it is a familiar task to many DH practitioners, though, so that may be something you want to spend time doing and documenting!

“But just kind of think about your process and how this will fit if you are planning the longevity of your project and how this can be one of the tools that you are using. “

This is the one that I did:

<https://voyant-tools.org/?corpus=f3fe6736cfbe158a6d8827f6c847b345>

**GitHub** (video tutorial by Jessica Dussault)

Before watching the tutorial, here are a couple definitions that may help you out…

* **Git** : software for version control, saves snapshots of files in time and keeps track of history
* **GitHub** : a website which creates a user-friendly interface for Git
  + Provides extra, non-Git features like issues, milestones, and team organization
  + There are many similar websites out there, such as GitLab (UNL offers a [GitLab account for students](https://git.unl.edu/users/sign_in)
  + [Links to an external site.](https://git.unl.edu/users/sign_in)
  + ) which provide a nice frontend for Git
* **GitHub Desktop** : software based on Git and meant to be used with GitHub, the website, which helps you work on projects locally
* Issues -question or bugs that have been found (other people can chat and comment on the problem)
  + There are labels
  + You can filter by author
  + There are milestone
  + You can customize milestones and labels
* Projects tab- you can create a to-do, in progress, review, and done. Helps organize what needs to be done.

How to create a repository

* Repository-
  + Click button on top right for new repository
* Create markdown files (format texts)
  + Edit to create lists, header, descriptions/paragraph
  + Use the cheat sheet

GitHub Desktop:

* You can make changes
* Download rep to desktop
* You can open it later on– and use a text editor like sublime editor
  + Save changes
  + And go back to GH desktop
  + You can see the changes
* Push origin: go back to GH to see the changes!
  + Easier to work with collaborators push and pull the changes that other people have been adding.
  + Always fetch origin when you start GH