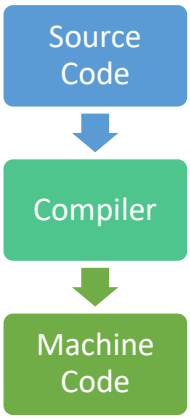
Using Open Source Software for Digital Preservation

- This section will introduce Open Source software and how it can be used for digital preservation.
- This will include the history and ethos of OSS, the pros and cons of using this type of software and information on how to get started using OSS.

Software 101

- Written in a human-readable programming language
- Most often 'Compiled' using an intermediary program into computer-readable form
- Proprietary software provides only compiled version
  - Can't make modifications beyond program's inbuilt functionality

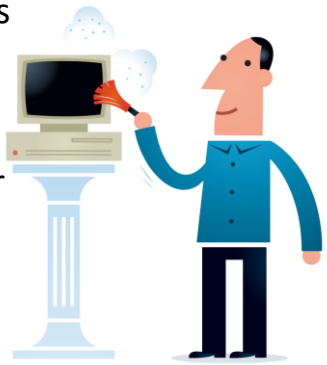


2

- To understand some of the issues and benefits of Open Source Software it is important to first be familiar with the basics of how software works.
- Most computer programs are written in a human-readable programming language such as Java or C. These contain complicated series of instructions for the computer to carry-out.
- These programs are not, however, understandable by a computer's hardware and so an intermediary translation must happen. This can happen in 2 ways:
  - Programs that are written in 'Interpreted languages' are parsed action by action as the program is run and the translations are supplied to the computer hardware.
  - Alternatively the program is passed through an intermediary programme called a compiler to translate the complete program into machine readable code. This will be the example we will use in this presentation.
- Most proprietary software is supplied in its compiled form.
  - As this is not human-readable it makes it virtually impossible to understand and alter.

History of OSS

- First conceived in late 1990s
- Adopt best practices from Free and Commercial Software
- Open development = better software
- First program released as OSS: Netscape browser
- Server/software infrastructure early priorities



3

- The concept of Open Source Software (OSS) was first introduced in the late 1990s as an evolution of the Free Software movement.
- It was created with the idea of adopting the best practices from both Free and Commercial software development.
- They hoped to retain the superior open development model of Free Software which had been proven to produce better software.
- This would be couched it in a more structured (but open) legal framework.
- The first program to be released as OSS was Netscape's browser, the code for which has since become the basis for the development of several other OS browsers including Mozilla's Firefox.
- Early efforts in the OSS domain focused mostly on server and software infrastructure projects but has since expanded to include all forms of software.

Ethos of OSS

"software should be made universally available in its entirety, with everyone afforded the opportunity to understand, change and re-distribute it"

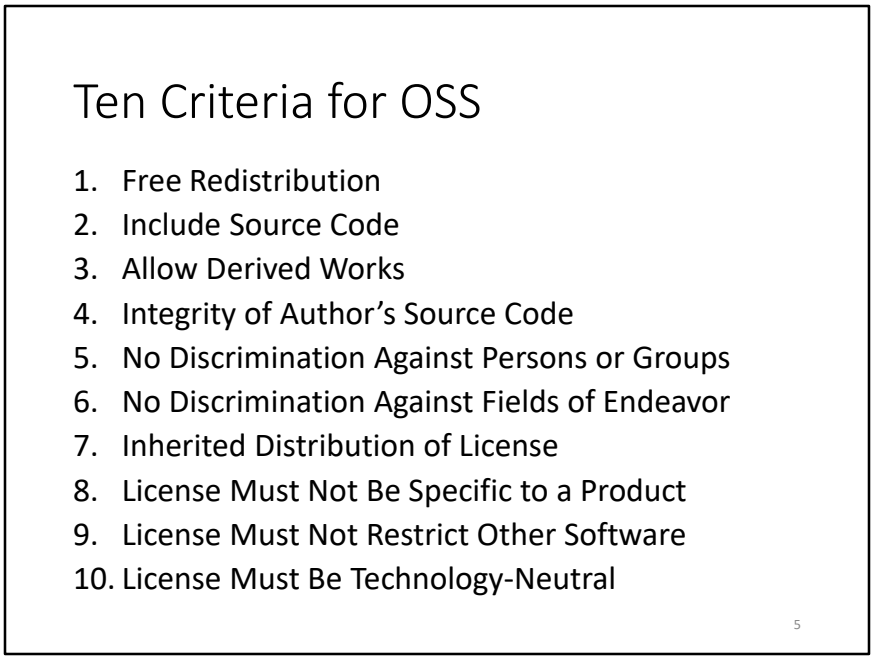
Andrew McHugh, DCC Manual, 2005

Key Elements of OSS:

- Transparency
- Openness
- Community

4

- The ethos of the OSS movement has been summarised well by Andrew McHugh in his chapter on the subject for the Digital Curation Centre's Manual.
- The OSS Movement believe in the ethos that: "software should be made universally available in its entirety, with everyone afforded the opportunity to understand, change and re-distribute it".
- So, key to OSS are:
    - **Transparency** – Making development process and decision-making transparent for all stakeholders.
    - **Openness** – The source code for all OSS should be open, likewise development should be open to all those wishing to participate.
    - **Community** – Fostering a strong and engaged community will create the best products.



As well as the general ethos of OSS, there are 10 key criteria that a product must adhere to to be considered Open Source. They are:

1. The product's license must allow for free redistribution, under the same license conditions.
2. All releases must include full access to the original source code.
3. Licenses must allow derived works, allowing users to customise software to their own needs.
4. Restrictions can only be placed on the redistribution of altered source code if the license allows the alternatives of the redistribution of:
    - The original source code with patch files, or
    - The altered source code with a different name or version number.
5. OSS cannot discriminate against any persons or groups
6. It can also not discriminate against any fields of endeavor, common examples being businesses or controversial domains such as genetic research
7. The terms of the original license are inherited by all who use a redistributed version of the product, this is sometime referred to as a viral license
8. If the product is part of a bigger package with other software the original license remains relevant even if the individual software is redistributed separately
9. The license cannot restrict what other software the product may be used with

10. The license cannot dictate the use of a particular technology or interface

A Free Beer, A Free Cat, or Free Speech?

A Free Beer

- OSS is not necessarily free as in 'gratis'

A Free Cat

- Costs relating to implementation, upkeep, training, support, etc.

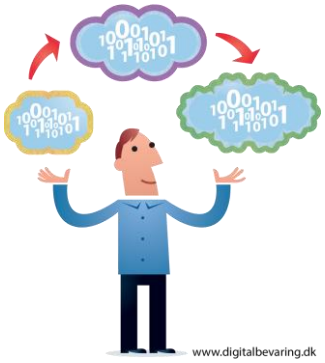
Free Speech

- Access to source code
- Ability to adapt to own needs
- Can redistribute

6

- Freedom and openness are key to OSS but many mistake this to mean that the software should be available free of charge.
- This is not the case and the freedom of OSS is often expressed using the analogies of 'a free beer', 'a free cat' and 'free speech'.
- If you receive a free beer, this is something that comes to you at no cost and you can consume without any further ramifications other than slight inebriation.
    - This is not the type 'free' that applies to OSS. It is often offered for no or a low cost but there is no requirement to be free as in 'gratis'.
- Some have likened OSS instead to the idea of a free cat; while the original gift may not cost you anything, caring for the cat will cost money for food, toys, vet bills etc.
    - With OSS although the original software may be free or relatively cheap, you will likely incur costs relating to implementation, upkeep, training, support and other issues.
- The other essential freedom of OSS has been likened to free speech in that there is a requirement for free access to the source code, to adapt the software if desired and to freely redistribute the product.

Development Model



- Users as co-developers
- Early releases
- Frequent integration
- Different versions: beta vs stable
- High modularization
- Dynamic decision-making

7

- There are several key ways in which the development of OSS differs from commercial solutions, these all aimed at creating more complete and stable products.
- The differences include:
    - Users are considered to be co-developers alongside programmers. This emphasises both the collaborative nature of OSS as well as the belief that testing and bug identification are as important to the development process as writing code.
    - Programmers are encouraged to release code as early as possible to allow the interaction described in the previous point, users can check functionality is fit for purpose and spot bugs early. This input leads to more productive development cycles.
    - Multiple programmers may be working independently on the product so they are encouraged to frequently integrate their work to ensure consistency.
    - The creation of modularized products which make it easier for multiple people to work on the product as well as enabling customization and updates.
    - Dynamic-decision-making is encouraged to ensure changes can be

incorporated quickly.

Different Types of Contributions

"Give as you can"

Help with:

- Scoping developments
- Identifying requirements
- Writing code
- Providing feedback
- Identifying Bugs

8

- As mentioned in the previous slide, the term 'developers' is used quite widely in the OSS world, including more than just those creating code.
- This is an important factor to remember if you are planning to use OSS but do not have the skills or resources to contribute to the programming efforts.
- Contributions are encouraged on a "give as you can" basis and all types are equally valued.
- These can include helping:
    - Make suggestions for and scope new developments
    - Identifying the more details requirements for developments
    - Contribute to the writing of code
    - Providing feedback on new functionality to make sure it is fit for purpose
    - Identifying bugs early to create more stable software
- Even a small amount of time spent on one of these activities helps the community at large.

SPRUCE Project

- Community orientated approach to digital preservation
- Collaboration on tools and resources
- Held 3 Mashups and 1 Hackathon
- SPRUCE Mashup Manifesto
  - Be agile
  - Re-use, don't reinvent the wheel
  - Keep it small, keep it simple
  - Make it easy to use, build on, re-purpose and ultimately, maintain
  - Share outputs, exchange knowledge, learn from each other

9

- The community orientated approach of OSS complements the approach many take to digital preservation and an excellent example of this was the SPRUCE project.
- It used a similar ethos to bring practitioners together to collaborate on the development of various tools and resources.
- These efforts included 3 'Mashups' and 1 'Hackathon' where DP practitioners and developers came together to work on small scale solutions to a variety of practical digital preservation problems.
- Like many similar OSS endeavors the project had a manifesto that encouraged participants to:
  - Be agile in their developments
  - Re-use existing code or solutions where possible so they were not reinventing the wheel
  - To keep things small and simple, approaching problems at a more atomic level with the idea that tools could be used together for more complex issues
  - Create tools that were easy to used, build on, re-purpose and ultimately, maintain
  - Share outputs and exchange knowledge between different development groups so they could learn from each other.

- These are all generally great points to remember for those getting started with digital preservation.
- All of the outputs of the SPRUCE project can now be found on the project pages hosted by the Open Planets Foundation.

Some Major OS Organizations

- Open Source Initiative
- Apache Foundation
- Mozilla
- Linux Foundation
- Free Software Foundation
- WordPress

10

- OSS is more widely used than many realise and a large number of organizations exist to oversee various processes and products.
- The Open Source Initiative is the original Open Source organization and the key body for oversight of OSS. They set the requirements for what can be considered OSS and approve licenses.
- The Apache Foundation maintains a large number of products that include the Apache HTTP Server, the market leading server software. They also now maintain the Open Office suite.
- Mozilla offer a range of Open Source Internet-related products including their most well-known output, the Firefox browser.
- The Linux Foundation also oversees a number of projects, the most famous of which is open source operating system Linux. Linux has been adopted by a wide variety of organisations around the world including several government bodies and the world's biggest financial exchanges, the NASDAQ and the London, and Tokyo Stock Exchanges.
- The Free Software Foundation was the precursor to the Open Source Initiative and has a more socio-political focus. It provides the framework to support the GNU project which, among other things, is responsible for one of the most common open source licenses.

- Another high-profile example of OSS is the WordPress blogging platform. Users can pay to have a blog hosted by the company or they can download the blogging software for free through an open source agreement and install it on their own web space.
- Open Source solutions have also been adopted and are distributed by a number of large commercial companies including IBM, Oracle and Google.

Benefits/Opportunities

- Likely to be lower cost
- More freedom
- Influence new tools/functionality
- Fewer license restrictions
- Improved debugging
- Builds communities
- Easier to emulate
- Can share tools with data creators

11

If you are considering using OSS it useful to be aware of the potential benefits and opportunities it provides as well as the risks and constraints. Benefits and opportunities include:

- **Lower costs** – Many of the OSS packages and tools for digital preservation are available free of charge. There may be costs involved in relation to implementation, upkeep and support but it has generally been found that OSS costs less in the long term than vendor solutions.
- **More freedom** – As already stated freedom to access, alter and redistribute is key to OSS.
- **Influencing new tools/functionality** – participation in the community surrounding an OSS project allows users to have direct influence on how it is developed.
- **Fewer license restrictions** – Licenses for OSS are far more open and through clauses like free redistribution can significantly reduce costs.
- **Improved debugging** – As debugging is a key part of the development process, with users actively contributing, this tends to lead to stable releases with far fewer bugs.
- **Builds communities** – The creation of communities around the development and use of the software provides users with a peer network to provide support and discuss implementations.

- **Easier to emulate** – key to digital preservation, the availability to access the original source code means that it is easier to emulate the original software environment if the program itself becomes obsolete.
- **Can share tools with data creators** – As well as only needing one license for multiple users, OSS allows you to share tools with data creators inside and outside of your organisation. This can help with preparing data before it is ingested in to your repository.

Risks/Constraints

- Tech resources/skills needed
- Lack of clear leadership and governance
- Requires community engagement
- Variable documentation
- Misconception about costs
- Securing institutional buy-in
- Potentially less diversity
- Too much customisation
- Funding/sustainability

12

Although there are many benefits and opportunities to the use of OSS, is essential to be aware of the potential risks and constraints so that you can take steps to mitigate them. They include:

- **Skills/resources needed** - OSS may need more technical skills and/or resources to allow its implementation that commercial products. This may be as simple as using a few command line operations through to full programming skills.
- **Leadership/governance** - sometimes OSS development suffers from the lack of clear governance and leadership which can result in the software becoming unfocussed or stagnating. This is mitigated by the existence of an identified owner or groups of owners to oversee decision making.
- **Requires community engagement** – without community engagement OSS may fail to meet requirements or find and sustain an audience.
- **Variable documentation** – The quality of documentation accompanying OSS has at times been poor, although this has generally improved in recent years.
- **Misconception about costs** – Too common and opposite misconceptions often exist about the costs of OSS. The first being that is free, the second that a large amount of resources are needed to maintain it without vendor support. The truth being somewhere in the middle, but explanation of true costs may be needed to get permission to use OSS.

- **Institutional buy-in** – it can often be difficult to secure institutional buy-in as those in management will often believe that OSS is inherently unstable and therefore presents a risk. Being prepared to advocate for OSS is important.
- **Less diversity** – Too much investment from the community in a single solution can sometimes lead to negative outcomes, reducing competition and diversity.
- **Too much customisation** – Customising OSS too much can move the software too far away from the main development and leave you without community support. Keeping customisations to atomic add-ons where possible is advised.
- **Funding/sustainability** – Some OSS projects suffer from a lack of funding and planning for sustainability. This has been true in relation to a number of tools developed by digital preservation projects. Once the project funding is finished the tool is no longer maintained. It is therefore information to check on these issues before committing to a particular solution.

OSS Licenses

- ‘Copyleft’ licenses
- Approved by OSI
- Emphasis on collaboration, openness and reuse
- Derived works must have same license
- Popular licenses include:
  - Apache License 2.0
  - GNU General Public or Library General Public Licenses
  - BSD 3-Clause or 2-Clause Licenses
  - Mozilla Public License

13

- The types of license used with OSS are often referred to as ‘copyleft’ as their emphasis is on providing a framework for freedom of use rather than focusing primarily on restrictions.
- To be accepted as a true Open Source license it must be approved by the Open Source Initiative. They maintain a list of approved licenses on their site, which also includes information on the most commonly used.
- OSS licenses are written to encourage collaboration, openness and reuse, with proper attribution to the original creators one of the few restrictions on reuse and redistribution. They are usually far simpler than their commercial cousins and a fraction of the length.
- They also require that all derived works adhere to the same license. Ensuring the ethos of OSS is passed to those works.
- Although there a large number of custom OSS licenses, many projects choose to use one of the more common standard licenses listed here. More information on these can be found on the Open Source Initiative website.

Comparison with Vendor Solutions

Issue	OSS	Vendor
Initial Cost		
Installation		
Source Code		
Customisation		
Licenses		
Bugs		
Support		
Documentation		
Training		
Motivation for Developments		
Succession		

14

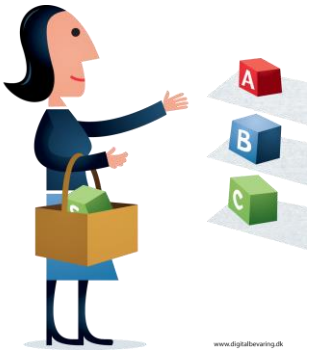
- The table on this slide shows a simplified visual comparison of using Open Source Software versus vendor provided solutions. Green = good in this area, yellow = mixed, some strengths and weaknesses, red = not available.
- Both have their strengths and weaknesses. Looking at these in a little more detail:
    - **Initial Cost** - Much OSS is free but even if there is an initial cost in procuring OSS it likely to be small in comparison with vendor solutions which may require a significant investment as well as an ongoing commitment to additional services or updates.
    - **Installation** – Vendors are likely to provide support with the installation of more complex pieces of software and simpler pieces are distributed in an executable format that is usually easy to install. Installation of OSS is more variable and may require an invest of resources and more technical skills to get it up an running.
    - **Source Code** – OSS provides access to the original source code whereas vendor supplied solutions are pre-compiled.
    - **Customisation** – By having access to the original source code, this means it is possible to fully customise OSS for your organization if you have sufficient programming skills. There is also the potential for greater customisation and collaboration from the wider community. Customization of vendor

solutions is usually limited to the in-programme options and tools. Any more significant customisation will depend on the vendor and their priorities. Some vendors will create customised modules for their software for a fee.

- **Licenses** – It is generally only to acquire only one license for OSS no matter how many installations are needed. They are also more open to the creation of derivative works and redistribution. Vendor licenses tend to be far more restrictive, limiting how and where the software can be used. It also normal that multiple licenses may need to be purchased one for each user or installation of the software.
- **Bugs** – Due to the collaborative nature of OSS development stable versions of the software tend to be less buggy and when bugs are identified they are addressed more promptly if a reasonable-sized community exists for the product. Vendor software tends to be more buggy when first released as getting the product on the market is a key. They may also be slower to address identified bugs later depending on their current commercial and/or development priorities.
- **Support** – Vendor software often comes with a support package, or this can be purchased as an addition. Meaning that there is some expectation of good and prompt support. The situation is more mixed for OSS software and depends on factors such as the size and engagement of the user community or the availability of paid-for support services.
- **Documentation** – Documentation was historically poor for OSS but the situation is much improved in recent years but it cannot be relied upon for all software. For vendor solutions, there is a reasonable expectation that good documentation will be provided when the product is procured.
- **Training** – Like documentation, training for OSS is mixed and sometimes only available for larger/more established programmes. Commercial vendors will more likely have training resources available. Depending on the size and complexity of the software this may anything from online resources to the provision of in-house training for staff.
- **Motivation for Developments** – One of the key strengths of OSS is that developments are normally motivated directly by the needs of the user community. The main motivations for vendors are usually focused on commercial concerns; such as making a profit and strengthening their position in the market. This can mean they are less responsive to user needs and will adhere to business models such as planned obsolescence.
- **Succession** – No matter the type of solution chosen it is important to carry out succession planning to make sure data can be retrieved in the event of the discontinuation of the solution. With OSS the continued support and development of a product relies on the ongoing engagement of the community. If this abruptly ends, access to the source code means users

are in a strong position as long as they have the skills/resources required. With vendors, it is very important to include succession planning in any service agreements but issues may still occur in cases on bankruptcy.

Things to Consider When Selecting OSS



- Longevity
- Stability
- Costs
- Ubiquity
- Skills required
- Documentation/training
- Compatibility

15

There are lots of factors to consider when choosing an OSS solution or tool, but a short checklist of issues might include:

- **Longevity** – How long has the software been available? Does it have a robust and active community of support?
- **Stability** – Do user comments indicate that the software buggy? Has a stable version been introduced?
- **Costs** – Is there a purchase cost? What will be the costs for implementation? Will you need to pay for support?
- **Ubiquity** – Is the software used by similar organisations? Can you rely on peer to peer support?
- **Skills required** – Do you have the necessary skills required to implement and use the software? If not, would they be easy to acquire?
- **Documentation/training** – Is the software supported by good documentation? Are there training resources available?
- **Compatibility** – Is the software compatible with your systems and other solutions or tools you have or would like to implement?

Beta vs Stable

Beta

- Version for community testing
- More bugs
- Latest features
- More updates

Stable

- Thoroughly tested
- Less buggy
- May lack new features
- Security updates

16

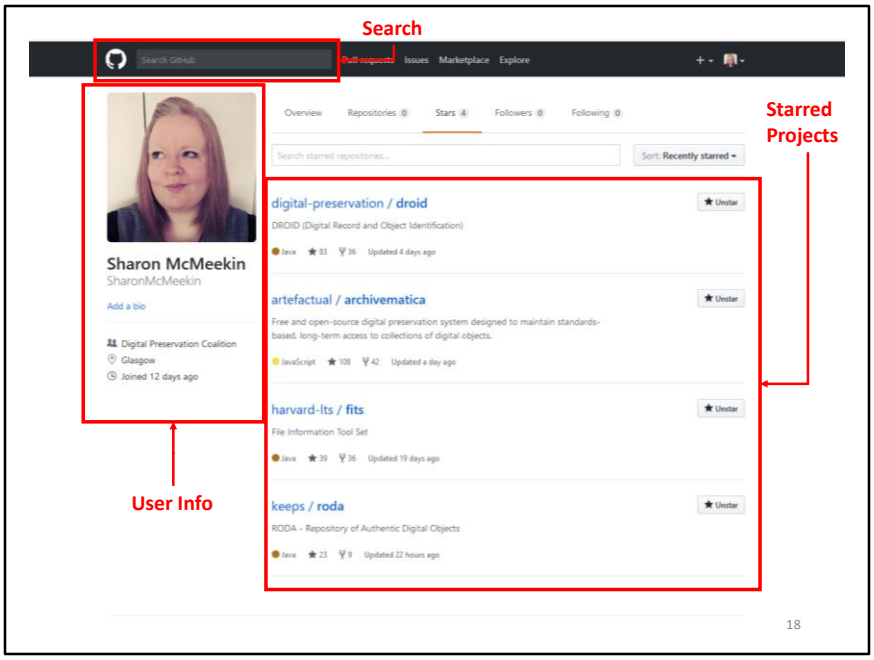
- One last important decision you may need to make when using OSS is whether to implement a Beta or Stable version of the software. Both choices have their positives and negatives.
- Beta versions are early releases of software that provide users with early access to new features so they can be tested by the community.
- You will have quicker access to new and potentially useful functionality but the software is also likely to have more bugs.
- This will mean that you will probably need to make more frequent updates to the version you are using (usually c. weekly).
- If you want to actively contribute the development of the software it is usually the beta version that you should use.
- The stable version has generally been thoroughly tested and more robust and less buggy.
- But it is likely to lack the latest features and, if the user community is not active enough, it may be a while before these are released.
- Stable versions generally require fewer updates (c. monthly or less frequent) and these are generally to fix security issues identified between versions.

GitHub

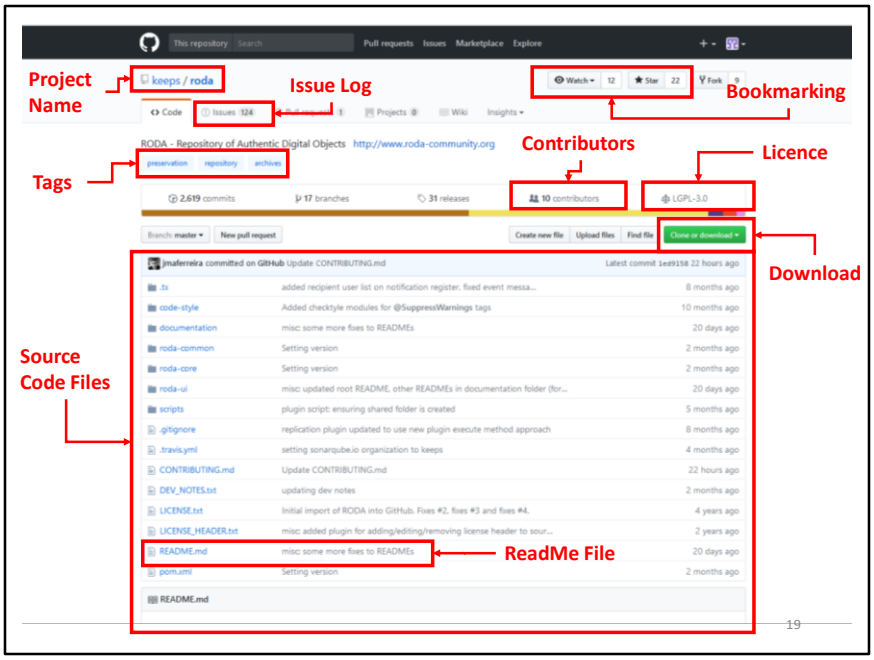
- A code hosting platform
  - Collaboration
  - Version Control (Git)
- Used by developers of the majority of OSS digital preservation tools and solutions
- Public and private development spaces
  - Basic account = free
- Access to full source code
- Best way to contribute to software development

17

- GitHub is a platform for hosting software source code.
- It allows developers to collaborate on the creation of software no matter their location and to managed version control through GitHub's Git solution.
- GitHub is the most popular online code hosting platform and is used by the majority of digital preservation-related OSS development.
- GitHub provides spaces for both public and private developments and basic accounts, which allow participation in public projects, are free.
- A project's 'respository' will provide full access to the software's source code. The repository is the name used by GitHub for a particular project.
- Interacting with developers on GitHub is the best way to help identify bugs and make suggestions for new functionality.



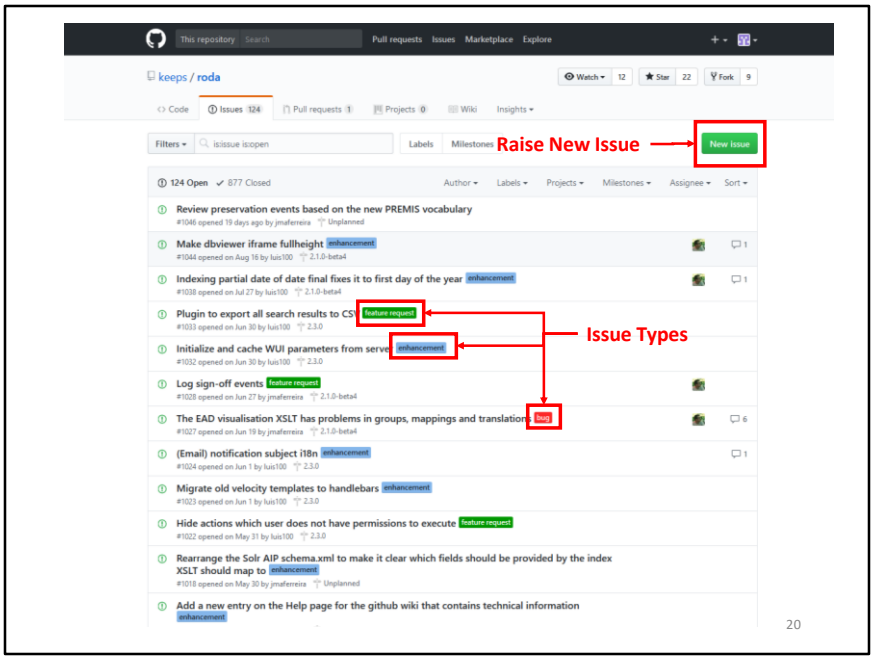
- The first step of participating in GitHub is signing up for a user account.
- As mentioned before, a basic account is free and paid upgrades are only required if you want functionality such as hosting a private development space or more advanced permission controls.
- Having a GitHub account will allow you to:
    - Add information about yourself and interact with other users in a 'social media-style'
    - 'Star' (or bookmark) projects you are interested in. The repositories for projects are named by the owner (an organisation or individual) and the project name.
    - Begin contributing to developments
- You can also see here the search box which will allow you to search for projects of interest.



- The GitHub Repository Page is the main portal for accessing information on a development project.
- Some important parts of the page include:
  - In the top left the **Project Name**, this has the owner (an individual or organisation) and the specific project.
  - **Tags** providing information on the type of project which can also be clicked to find similar projects.
  - All of the **Source Code Files and Supporting Documentation**, this should always include a **ReadMe** file which should provide an introduction to the software and project. This will usually include a link to where you can download an executable version of the software (if available).
  - The files can all be downloaded individually or there is a button to allow you to **Download** a complete version.
  - There is a section that contains details of the corresponding **Licence** for the software.
  - You can also see a list of **Contributors** to the project, including data on when they contributed and how much.
  - At the top right, there are buttons to facilitate **Bookmarking** a project. Choosing to 'Star' a project will added it to your 'Starred' projects list.

'Watching' a project will notify you of any conversations around the project. This useful for projects you plan to actively participate in.

- Finally, there is a tab for the **Issue Log** this is the primary place where non-programmers can contribute to a project.



- The Issue log allows project contributors to raise issues they have found with the software.
- This can include bugs that have been identified, suggestions for improvements in performance and suggestions for new features.
- A well-structured issue log will have the issues tagged by type to help the developers identify priorities.
- There is a button in the top right of the page to allow participants to raise new issues.
- As previously stated, testing code and providing feedback is essential to the successful development of OSS and so contributions to the Issue Log are both important and welcomed.

Types of OSS for DP

- Two main types of open source for digital preservation
- Large-scale applications
  - Repository systems
  - Storage
  - Workflow
- Tools for particular functions
  - Characterisation
  - Migration
  - De-duplication
  - …

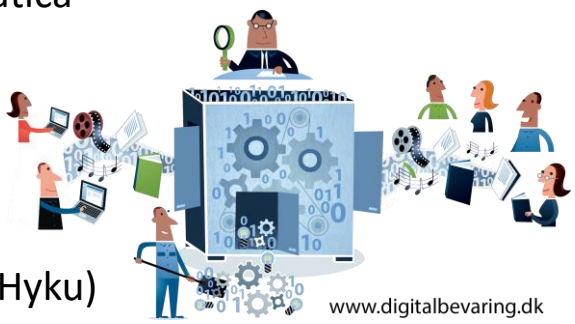
21

- When looking at OSS software for digital preservation there are two main types of product you may consider using.
- The first are large scale applications which can be used to manage multiple processes.
- These can include complete repository systems, software for managing storage and workflow management systems for implementing potentially complex process.
- The other main type of OSS for digital preservation is smaller tools that carry out particular functions, which can be smaller-scale processes or a step in larger processes.
- These can include tools for characterising a digital collection, for migrating a particular file type or to check a folder for duplicate file.
- There are many of these smaller tools and often several that will carry out the same function.

Example Repository Systems

OSS repository systems incl.:

- Archivematica
- RODA
- DSpace
- Fedora
- Eprints
- Samvera (Hyku)



22

- If you wish to implement a full repository system for your digital collections, there are an increasing number available.
- Archivematica and RODA are two examples of repositories that are well supported by both a specific organisation and their user community. Both system offer free repository solutions with additional plug-ins and paid-for support services.
- DSpace, Fedora and Eprints have emerged from and generally been used more in the research data and publication domains but have been implemented by a variety of different organisations.
- Samvera is a repository solution that has evolved from the Hydra project and collaborative work of a number of Higher Education institutions. Their Hyku solution aims to be an easy to install ‘out of the box’ repository.

Example Tools: Characterisation

Various tools with different functionality:

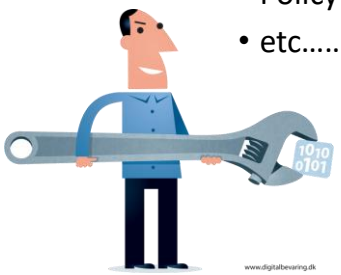
- DROID
- Apache Tika
- C3PO
- FIDO
- JHOVE
- FITS

23

- One of the most widely used types of OSS tools for digital preservation are those used for characterisation, and several are available.
- DROID is developed by The National Archives of the United Kingdom and links to their PRONOM database of file format information. It is one of the most widely used characterisation tools as it is available with a graphical user interface and includes functionality such as fixity checking.
- Apache Tika, C3PO, FIDO and JHOVE all offer similar functionality but with different strengths and weaknesses. For example, JHOVE provides the richest output including file format validity but only for a limited number of format types.
- FITS (the File Information Tool Set) is a little different from the other tools as it actually packages together a number of the other characterisation tools including DROID, Apache Tika and JHOVE. This means it can produce rich results but also inconsistencies between the different tools.

Other Types of Tools

- De-duplication
- Forensics
- Decryption
- Fixity
- Planning
- Migration
- Emulation
- Validation
- Policy
- etc……

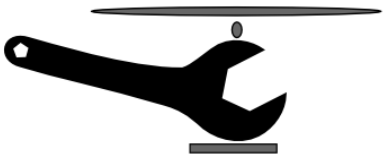


24

- There are many other types of OSS tools for digital preservation such as tools for:
  - Identifying duplicate files
  - Carrying out forensic analysis of files or directories (particularly useful for handling processes such as disk imaging as well working with protected or sensitive files)
  - Decrypting files
  - Checking file integrity using fixity values
  - Carrying out preservation planning
  - Migrating file formats
  - Accessing file using an emulator
  - Validating a file format matches the format specification
  - Helping to write policy
  - And many other tasks and processes….

COPTR

- Tools registry for digital preservation
- Includes OSS and Vendor solutions
- Part of DigiPres Commons
- Hosted by the Open Preservation Foundation
- Browse by:
  - Name
  - Function
  - Type of content



25

- When looking for tools for digital preservation, one of the most useful places to start is the tools registry COPTR.
- The registry includes listings of both OSS and vendor software and solutions
- It is one of the information resources offered by the Digi Pres Commons and it hosted by the Open Preservation Foundation.
- COPTR allows you to browse tools by name, function and type of content.
- It is a community developed resource so the amount information varies by tool but contributions are welcomed.

POWRR Tool Grid

	DCC Lifecycle Stages*							
	Access, Use and Reuse	Create or Receive (Acquire)	Cross-Lifecycle Functions	Dispose	Ingest	Preservation Action	Preservation Planning	Store
Audio	2	5	3		15	11	1	
Binary Data			4					
Container						5		
Database	1	1	3		3	14		
Disk Image		7	4		3	1		1
Document	3	1	4		33	14		
EBook					5	2		
Email			5		2	4		1
Geospatial					1			
Image	2	2	3		23	23		
Project Management Data	1					1		
Research Data	2	8	13		4		16	17
Software		1	1		2	2		1
Spreadsheet					6	3		
Video	1	3	1		10	8	1	
Web	3	21	2		7	3	1	1
~Not Content Type Specific~	22	38	83	9	69	61	31	51

26

- The POWRR Project has been a major contributor to the COPTR repository and another eay to navigate the site is using the POWRR Tool Grid shown here.
- It allows users to identify relevant tools by object type and by lifecycle stages, which is particularly useful when developing new processes.