Onboarding Digital Humanities Students with a Shared Working Environment for Introductory Courses: Concept, Implementation, and Lessons Learned

Claus-Michael Schlesinger¹ Malte Gäckle-Heckelen
Fabienne Burkard¹

¹University of Stuttgart

Published on: Jun 06, 2023
DOI: https://doi.org/10.21428/f1f23564.4c0566bc
License: Creative Commons Attribution 4.0 International License (CC-BY 4.0)
In our DH teaching and learning practice, the basic use of a computer for research purposes was often taken for granted. This assumes that students have the skills to administer an operating system, to install and administer new software (like integrated development environments, off-the-shelf software, and special purpose research software), and sometimes even to use a command line interface (CLI). However, students in the introductory courses at the beginning of the DH MA program at the University of Stuttgart (Germany) come from different disciplinary backgrounds and start with different levels of expertise regarding the uses and configurations of computer systems. In our courses, we have identified the need for a consistent and accessible computational workspace in order to minimize configuration support and software problems and create an environment for a productive and inclusive learning experience. To answer this need, we devised DH2go, a system based on the free and open-source remote desktop software X2Go (X2Go development team). As of now, students in our introduction module only have to install X2Go in order to gain access to DH2go, a consistent Linux system with a graphical user interface (GUI) and all necessary DH tools and data pre-installed. In this article, we describe the initial set of problems we faced when working with students in introductory DH courses, the system we designed in order to solve these problems, and our design process. We think that our problem analysis and our approach, though based on a specific use case, might be applicable in other DH contexts and inform critical reflection about the role of digital infrastructure for DH courses and DH pedagogy.

The introduction module in the Stuttgart DH MA program spans several technologies, with a focus on text analytics and text modelling. The module contains a lecture for DH theory, history, and methodology basics, and a course for practical application of DH methods. The course is centred on competence-oriented teaching and learning with a strong hands-on component. Due to the wide spectrum of DH methods presented in the lecture and applied in the course, time is limited to roughly two sessions for each topic. Students bring their own devices, and in our own teaching experience, it soon became clear that system configuration tasks could become a major issue for many students, leading to time-consuming preparation and individual or collective support sessions. Course preparations included installing an integrated development environment or code editor for the Python programming language along with the extra packages needed for specific tasks, installing Java software, and finding a CLI for the respective operating system. In some cases, installation problems could not be solved either by students while preparing for the session or in session, meaning students were not able to participate fully.

For courses and pedagogic concepts with a strong practical component, design and configuration of the working environment plays an important role. Making the configuration and set-up of such an environment a task that students need to fulfill before and outside of the course is not feasible in some cases because students new to DH often lack the necessary skills and routines. This leads to frustration and a high demand for support which teachers cannot provide due to time constraints and a lack of expertise—who knows how to install, configure, and troubleshoot software packages on all flavours of Windows and MacOS and on various Linux distributions?
As the introduction course is focused on the application of DH methods, a consistent working environment is a precondition for a fulfilling and productive learning experience. However, this leads to a circular problem: The configuration of a reliable computational working environment requires knowledge of basic workflows in computational research, while learning these workflows and developing computational competence requires a consistent and well-configured working environment. DH2go is a server-based teaching and learning environment for DH courses and workshops with a strong practical component. It has been in productive use at the DH department at the University of Stuttgart since 2018 and is under active development. It provides a virtual desktop and shell access, and all software used in the courses is pre-installed. The system is configured as a multi-user and multi-course system, so students and teachers may use the system for different courses at the same time.

In DH pedagogy, practice and learning-by-doing approaches play an important role. E. Leigh Bonds (149) identifies “collaboration” and “project-based learning” as core concepts in contemporary DH pedagogy discourse. An additional component in DH teaching is often a playful and explorative approach to objects, research questions, and data (Tracy and Hoeim; Hirsch 15). However, the value of playful work with its focus on process and alternative and surprising solutions to problems is difficult to communicate, especially in regard to graded results, where grades do not take into account process and learning experience (Tracy and Hoeim).\footnote{1}

This playful communication and collaboration is further impacted by the often multifaceted forms of teaching technology and environments, as these determine the reference points students use to orient themselves and others in their new learning space. From this perspective, digital infrastructure is part of the socio-technological entanglements that shape interactions and subjectivities in teaching and learning situations.

Although playing and tinkering with data and software might seem to be possible without much preparation, this is not the case. Playful sessions and tinkering assignments require preparation, a well-configured system, and basic exploration skills. If exploring alternative approaches becomes time-consuming due to technical difficulties or lack of experience, the incentive for exploration will be diminished. In this case, even playing, tinkering, and experimenting becomes a question of access and exclusion. If technological preconditions—this can be hardware as well as technical experience—cannot be met by students, experimenting either will not take place or it will not have the intended learning effect (Moisich). This is even more true for collaborative learning scenarios, as different systems might offer different paths, possibilities, and error messages that need to be taken into account when working on a collective solution to a problem.

Oliver Moisich sees the role of digital tools as enabling students and teachers to “access digital approaches and to start appreciating their relevance for research.” This view on software for DH teaching shifts the focus from tool capability to accessibility: software needs to be accessible for students with different requirements based on hardware, operating system, and technical expertise.\footnote{2}
Use Case: DH Introduction Course Concept and Practice

To understand why we chose a certain way to design DH2go, a look at the context of its inception is important. Students in the DH MA program at the University of Stuttgart start with a background in (almost) any humanities discipline, with media studies, sociology, computer science, and natural language processing (NLP) being rare exceptions. Most students have not been in contact with any structured educational courses regarding computer science or statistics.

The Stuttgart DH program curriculum addresses this necessity to introduce students to basic concepts in computational analytics and research in a rather forward manner: a wealth of technical knowledge is taught in the first semester, while specialization occurs in the following semesters. The first semester DH modules make up only part of the curriculum, while other modules are focused on advanced study in humanities domains. These are oriented on technology and methodology. They consist of a programming course with an introduction to the Python programming language, a module with an introduction to NLP methods, and the DH introduction module. The second and third each include an overview lecture and a complementary course with practical exercises.

As the DH introduction module aims to provide a broad overview of DH theory and methodology, the practical course touches on several DH fields with an emphasis on text analytics and modelling:

- HTML
- CSS
- XML
- XML/TEI
- SQL
- PHP
- R/stylo and stylometric methodology
- Topic Modelling: LDA and Word Vector Models in Python
- Network Analysis with Gephi
- GIS

While most sessions build on concepts introduced earlier in the course or in other courses, sessions on Network Analysis and GIS feature concepts and technologies often entirely new to students. All in all, first year students have to deal with a high workload, lots of information, and a steep learning curve while developing computational competencies. Students need to make a substantial effort in order to integrate all course contents into a somewhat consistent whole. Starting in the second semester, students specialize: they may choose courses and seminars following their own interests and technological preferences. Students choose humanities, computer science, and DH seminars with a relation to DH methods and objects and develop their own research approaches in a project-oriented seminar in the second semester and in their MA thesis.
The project-oriented seminar, which can be connected to active and ongoing research projects in the university and to production and development projects at cultural institutions (museums and archives), offers students the chance to apply and strengthen their scientific, organizational, and collaborative skills to plan and execute a project or sub-project in a larger institutional context.

In our experience, students usually struggle at the beginning of the first semester but make steep progress at the end of it. In the second semester, most students develop stronger confidence in their own skills and interests during courses and project work. With a broad frame of reference regarding DH theory and methods, most students will be able to seek out resources relevant to their problems.

However, teaching and learning in the introduction course also showed several systematic problems causing frustration for both teachers and learners, thus affecting the learning outcome in a negative way. A tight schedule focused on specific methods does not allow for the introduction and discussion of very basic aspects such as workspace organization and system configuration. We experienced a gap between the expectation that the basic tools—the computer systems used by the students—would just work without any issues and the reality of a wide range of hardware and operating systems, spanning from small consumer laptops with old versions of Windows or MacOS and old business laptops running Linux to top-notch machines fresh off the shelf. Taking such an ecosystem of computer systems and adding the necessity of installing yet another software package for each session in the course led to frustration and a lack of time for actual course work due to high support requirements:

- In the course, more time might be spent with installation and configuration than with discussing and trying out DH methods.
- Students with poor equipment and students lacking technical confidence need a lot of time for system configuration, or they might not be able to prepare their system at all. This leads to frustration and a low degree of participation, barring students from accessing DH methods and achieving the objectives of the course.
- Communicating problems can be difficult for students, as error messages can be hard to understand (and even hard to find).
- DH teachers are not experts in system configuration and might not be able to find a solution to problems.
- Students bring their own device containing not only data and programs related to learning, but also to their private life. A quick check for specific configuration files or software versions by staff thus leads to privacy concerns.

These negative effects cannot be easily generalized. However, we believe that they have to do with elements of the course concept, as discussed above:

- Overview course: Many technologies and concepts are featured.
- Variety of backgrounds: Participants have different technical backgrounds and set-ups.
Onboarding Digital Humanities Students with a Shared Working Environment for Introductory Courses: Concept, Implementation, and Lessons Learned

While different computer systems in a group can be a source for learning and exploration, they can also disrupt learning processes if problems arise that need to be solved before being able to work on specific assignments. The lack of common ground makes communication and support difficult. And while it is certainly important that students get to know their own computer systems and how to properly configure them, this process takes time.

Strategies for Digital Learning Environments

In order to minimize and streamline support and maximize accessibility, course participants’ systems need to become transparent for teachers and for support, and to a certain degree even for participants themselves. “Transparent” in this case means that support staff and teachers do not deal directly with participants’ systems in order to provide a consistent computational working environment. In some cases, participants use their systems to access a working environment, while technical access requirements are minimized, e.g., by providing access through a web browser. For our use case, transparency can be achieved through multiple strategies:

- Define a strong BYOD (“bring your own device”) policy, expecting from participants that they configure their systems properly in advance. This is what was initially anticipated in our use case and what led to the aforementioned problems.
- Roll out institutionally controlled systems (departmental laptops or computer pools) that are pre-configured for anticipated coursework. For our use case (15–25 students per year), the costs and maintenance requirements are high compared to the strategies listed above and below and might not be feasible for small to medium departments. Computer pools, on the other hand, might be established already at a university. With virtualization it is possible to run predefined configurations in a pool (Suchodoletz et al.).
- Provide a remote solution with a GUI. Examples for this strategy range from the use of various web services like Voyant Tools (Sinclair and Rockwell) via specific solutions like a collection of Jupyter notebooks (Karsdorp) to large service infrastructures like the Digital Humanities Virtual Laboratory (DHVLab) at the Ludwig Maximilian University of Munich (LMU Munich; Klinke) and more generic institutional remote desktop infrastructures (Mills). This strategy can be implemented by using an existing service, where maintenance and administration can be outsourced. However, outsourcing can also mean less control and using a design out of the box. Alternatively, running the service directly provides full control over design, configuration, and administration, but requires a substantial amount of time for planning and development.

For our use case, we considered and tested DHVLab, a service provided by LMU Munich. DHVLab is a rich infrastructure providing many services for DH researchers, learners, and instructors. Maintenance, system
configuration, and user administration are managed centrally by the provider. Specific software will be installed on request. A centralized approach necessitates the transfer of course data and user data which, for our use case, was not feasible due to concerns regarding institutional privacy regulations and the time constraints for feature requests. Because of evolving and often urgent needs with regard to course administration software packages and sensitive user data, we needed to be able to do user administration and software installation ourselves, and we could not rely on third-party management. We found one other solution centred on DH requirements, DH Box. The system is built around a docker container with high level user and administration access and installation scripts. It provides several basic components of a DH workstation (Ipython, RStudio, NLTK and Omeka) (Zweibel). Unfortunately, the project seems to be abandoned, as the last changes in the GitHub project date three years ago and the last tweet from the project’s twitter account was sent out in 2018.6

**Development Goals and Implementation**

DH2go, the system developed for our use case, aims to provide a complete workspace based on an operating system.2 We set five basic requirements which take into account the institutional situation in a small to medium-sized department regarding funding and staff. The system should be

- Stable
- Secure
- Sustainable
- Accessible for learners, teachers, support, and administrators
- Versatile and thus suitable for a range of courses

The system should be stable in order to minimize maintenance and troubleshooting. It needs to be secure for obvious reasons, as a learning environment should be a space where students and teachers do not need to worry about bad-faith intrusions. The system should be sustainable in a double sense: the home institution should be able to keep the system in production even when being developed and run by staff with a fixed-term contract. And it should be sustainable regarding efficiency and the use of natural resources. It should be accessible for all stakeholders, as accessibility for groups is interdependent. While accessibility for learners is most important, it relies on proper access for teachers, support, and administrators. The system should be versatile insofar as the basic use case’s requirements direct design decisions, while these design decisions should be feasible for a range of course and workshop types, especially on a basic level (see below for a detailed discussion of specific and generic qualities of the system).

These basic goals were translated into a more specific development policy. In order to reach a low level of complexity for sustainable development and production, we used a server-based approach with Debian Linux as the core component of the system. System administration relies heavily on mechanisms and programs provided by Debian Linux. The primary software management level is Debian package management. Specific Python packages that are not available through Debian package management, like Spacy, are managed using
Python package management. The same is true for R packages. Only if a software package is not available through any package manager is it okay to install it directly, but mostly we opt for an installation on the user-level in those cases. Regarding the catalogue of installed software, the most important directive is to only install software that is actually needed.

By applying a simple and well-defined system and user administration policy, administration and configuration become mostly reproducible. This is important for versatility and for supporting use cases other than the initial one. The complete solution should be portable so other teams and institutions can use it and modify it to their needs. Versatility is also the reason why DH2go is strictly based on free and open-source software (FOSS), the only exception being the Oxygen XML Editor, as it is widely used not only for scholarly digital editions but also in the publishing industry.

DH2go is primarily a remote desktop system, so apart from Debian Linux, the most important component is the remote desktop server solution X2Go. The X2Go client is the single software package that students need to install on their own systems. It is the gateway to the working environment on the server. In order to minimize the support effort necessary for onboarding a course into using the system, detailed tutorials provide step-by-step descriptions of how to install and configure X2Go on Windows, MacOS, and Linux systems (Burkard et al.).

With DH2go, an identical interface is used by all involved in the course to overcome the technical barriers mentioned above and to facilitate communication (not only between teachers and students but also peer-to-peer). In our experience, this has made onboarding easier for students and has furthered collaboration, as new groups get to know the system step by step together, discussing questions regarding system set-up and workflows more easily because they share a common reference. Additionally, peer-to-peer problem solving in hands-on situations is easier and more straightforward, as students and teachers alike can reference reproducible steps that need to be done in order to achieve a certain result.

System administration regarding rights management is user-centric and course-oriented, meaning users are able to configure their own workspace in terms of desktop design and are allowed to install additional software, e.g., Python packages, in their user space. Each course is reflected as a group on the operating system level using participant lists. By default, each course is provided with a folder for the distribution of material by teachers (course folder) and a folder for sharing data with everyone in the course (sharing folder). Both features help in providing structured data for exercises inside the working environment, so course participants start with an identical set-up but have the ability to move and change data in their own workspace as needed.

**Comparison**

In our research on available solutions, we found three other approaches similar to ours, all developed in response to installation issues: DHVLab, DH Box, and dedicated collections of Jupyter Notebooks, which can be made available with a central server approach using JupyterHub.
The DHVLab is a large infrastructure providing several web services and a remote-desktop solution intended as an environment for learning, teaching, and research. DHVLab and the remote desktop solution is being developed at LMU Munich (Klinke 29). As such, it does provide common software used in the DH on Linux-based virtual machines (CentOS), accessible via a standard web browser (Klinke 30). In addition to the remote desktop providing common Desktop DH tools like Gephi, DHVLab offers multiple web services, e.g., RStudio Server and the aforementioned JupyterHub. While DH2go features a minimal software stack, customizable to the requirements of specific courses, DHVLab provides a wide range of web services. DH2go is focused on learning and teaching, whereas DHVLab also offers services for scientific workgroups and small to medium research projects, including tools for research, publication, documentation, and research data management. DHVLab applies a strict open source policy and makes system information accessible for reuse. In principle, it is possible to implement all services on a self-hosted server cluster. However, providing interconnected web services and connections to additional critical services like long-term research data management necessitates a centralized and scalable infrastructure, whereas DH2go’s approach to versatility and reuse is to reduce complexity and enable departments, instructors, and students to configure, fully control, and host their own system adapted to their needs.

DH Box is a remote solution that offers technologies via a browser user interface. At this point in time, DH Box is not actively maintained, but all the files necessary to implement DH Box are still available via the project’s GitHub repository. We were not able to directly test DH Box, but Stephen Zweibel describes the project in detail. DH Box is deployed via virtual machines within a Docker framework (Zweibel 6). Users sign up on a project website and configure an instance to their liking, choosing among common DH tools which are then made accessible over an IP address (Zweibel 7). Users interact with tools via a user interface, choosing between the CLI, Jupyter Notebooks, MALLET, and many more. DH Box is versatile, as the configuration of users’ virtual machines is very flexible. While in spirit offering similar things, DH2go differs from this approach through its GUI and usage scenario: while DH Box is meant to be used as a remote toolbox, DH2go is more of an environment and offers a complete “workshed” in the form of a full-featured desktop environment. Users work on a full-fledged system that, in theory, they could use as the primary operating system for their local machine.

The (remote) desktop approach is also the key difference from web service solutions for programming environments like RStudio Server or Jupyter Notebooks deployed via JupyterHub. Here the focus is on programming, while DH2go also makes available dedicated software packages for data exploration and data analysis and a working environment similar to users’ regular desktop environments.

But if DH2go is based on the software X2Go, how is it different from setting up any kind of Linux server and using X2Go to let students work on it in a GUI? This is not uncommon at technical colleges, e.g., at the RWTH Aachen University, Germany (Baur). Well, that is exactly what we do! However, while DH2go is just another Linux server, configuring a system for specific learning and teaching purposes requires decisions regarding
design and use cases, policies and processes for user and course management, onboarding, and support strategies. Furthermore, DH2go is not only a service for DH seminars and workshops but also a framework for deploying a remote desktop environment for DH learning scenarios, comprising policies, tutorials, tested configurations, and helper scripts for automating administration tasks.

**Current Usage and Usage Scenarios**

Since its inception, the system has been used constantly in our DH introductory courses and, additionally, for specialized project seminars. In view of usage scenarios of the system, we differentiate between concrete qualities and configuration of the system and more basic generic qualities. Explicit configuration consists of default desktop configuration or the list of software packages provided to all new and active courses. Additionally, we consider the catering to the needs of specific courses as explicit configuration, as these mean direct changes or additions to the system.

The system can also be described in regard to its more generic qualities on a basic level of system configuration possibilities. The course-oriented development policy has been designed to cater to individual course needs. This means that DH2go always offers the basic minimum of software packages required for coursework, and additional packages will be installed on request. This course-oriented configuration policy leads to a less complex and more versatile basic system, keeping resources in computing, storage, and personnel free to accommodate new courses with specific requirements.

We consider these basic concepts and policies transferable to other fields, especially the initial situation and problem analysis regarding installation and configuration of software on participants’ computers and the time constraints in introductory courses with computer-aided components in small to medium departments. Also, this development and configuration policy, user management strategy, and basic support strategy might be transferable to other contexts similar to our initial use case. This would be, in our view, the case for DH workshops with a strong practical component, which have less time compared to a regular university course and therefore need to solve installation problems even more efficiently than teachers and participants in our initial use case. DH2go could, furthermore, be adapted to provide a working environment for small to medium work-groups and for small to medium learning scenarios with recurring tasks like training in libraries, archives, or museums or hackathons for teams with different backgrounds.

**Lessons Learned**

There are several things we would have liked to know before we started. Most important might be that building infrastructure requires that a lot of work be put into development, implementation, maintenance, and support. Building infrastructure also means a lot of responsibility. With this responsibility also comes greater power, independence, and potential for creative pedagogy. While not every scholar needs to be able to provide this infrastructure, we argue that scholarly involvement in the development of and critical reflection about digital infrastructures and their socio-technological entanglements might lead to better and more accessible designs.
However, providing users with a stable and secure system for their work and for their data is in itself a major task, even before we start learning DH methods using that system. Software and software-dependent systems need constant development and care. At the same time, a development and administration team with only fixed-term contracts means that the system is running on empty from the start and needs to be designed so that others can take over all tasks required for stable and secure production.

Support for users implies support for teachers. Onboarding processes are necessary not only for students, but also for teachers, as working with a new environment is a challenge even for seasoned DH professors and lecturers. Last but not least, DH2go development started on the department’s compute server. That server had a very complex set-up regarding web services and software packages. Only when we moved the part centred on learning and teaching to a dedicated virtual server were we able to focus on the specific requirements in our initial use case and develop policies for server administration, onboarding, and support.

A lesson that led to DH2go in its current iteration is that there need to be well-defined channels of communication for support. After initial tests of the system in our introductory courses, we found that having all those involved also function as support staff results in confusion and chaos on the system. Introducing a suite of tutorials for users, a two-level support system with only one contact person, and clear criteria for escalation as well as a changelog made the process more clean-cut and kept the system stable.

Conclusion

Installation problems are common in DH courses and workshops. So they were in our introductory courses. Installation problems require time consuming support and lead to situations where coursework becomes inaccessible to students. Installation problems are not only technical in nature but can be described as an effect of a socio-technological configuration, taking into account the respective backgrounds of students and instructors. From this perspective, installation problems become a basic question of accessibility that needs to be addressed accordingly. Multiple approaches to deal with installation problems exist. However, most solutions assume the use of a centralized infrastructure. In our use case, this would lead to user data management issues and a time constraint regarding versatile configuration of the learning environment according to specific course or workshop content.

Developing and running a dedicated server with a consistent and complete desktop environment providing all software packages necessary for a specific course or workshop can be a tailored solution to the installation problem. From the perspective of our initial use case this implies not only technical, but also—and more importantly—pedagogic and social aspects. Most importantly, onboarding new students and new groups relies on several facilitating elements: a well-structured introduction, detailed tutorials for non-experts, and an efficient support system.

In our experience, the most positive effect of groups working in a shared stable environment with reproducible workflows was that it improved collaboration due to a common reference system and furthered students’
confidence in their own computational competence and ability to deal with whatever problem arose when working with the system. This, we found, is a big step in an onboarding process for DH students. Taking into consideration the social dimension, the meaning of the term “teaching and learning environment” implies additional layers. These socio-technological entanglements are a key factor in DH research as in DH teaching and learning and need to be taken into account when developing, implementing, and running such an environment.

Works Cited


**Footnotes**

1. It seems that a core function of grades—to provide a means of comparing performance oriented on specifically defined criteria and to assess and sort results in a hierarchical fashion—can be counterproductive in activating and building on the intrinsic interests of students that, in the best sense of playfulness approaches, can transgress grading criteria. ⇐

2. By “accessibility,” we mean accessibility in relation to knowledge and competences. We understand that the term also implies questions of access and accessibility for persons with disabilities and that we need to address these questions before they arise in a concrete specific use case. ⇐

3. As of now, there are only a limited number of DH BA programs in Germany, and these have been established only recently, e.g., at the Technical University of Darmstadt and at the University of Wuppertal. ⇐

4. The author team for this paper consists of one participant in this introductory course and two of its instructors. ⇐

5. For more details on the DH program at the University of Stuttgart, see Nils Reiter et al. and Gabriel Viehhauser et al. ⇐
6. See the project’s [GitHub page](https://github.com) and the project’s [Twitter account](https://twitter.com). 

7. See the [DH2go website](https://dh2go.org) for an additional overview description of the system with technical details and our installation tutorials for X2Go client software (Burkard et al.). For future reference, we provide the current iteration of the website and the current version of our helper scripts for user and course administration via Zenodo (Schlesinger et al.). Both website and administration software are under active development. 

8. We provide installation information for the basic DH2go system on the DH2go website and desktop configuration files and helper scripts for user and course administration in a [Github repository](https://github.com). See Schlesinger et al. 