# Một Garage GitHub cho một số Nhân văn khóa học

Vào tháng Tám năm 2015, Pitt-Greensburg nhân văn kỹ thuật số giáo viên hướng dẫn thực hiện các DHClass-Hub, một kho lưu trữ GitHub thiết kế để sinh viên làm quen với GitHub qua sử dụng hàng ngày. Đây GitHub chức năng môi trường giống như mộthợp tác xã *gara* nơi học sinh là cả khách hàng và các cơ chế. Trong không gian kho họ chia sẻ mã từ bài tập và các dự án mà họ cần giúp gỡ rối, vì vậy mà họ và bạn học của họ và các giảng viên có thể kiểm tra, tư vấn về, và sửa chữa nó lại với nhau. Mặc dù DHClass-Hub được thành lập chủ yếu để giúp học sinh đạt được sự tự tin với GitHub để cải thiện quản lý các dự án học kỳ của mình, kho bất ngờ trở thành các locus của một kinh nghiệm học tập lật, nơi mà các hoạt động học tập sống động của khóa học mã hóa xảy ra trong đối thoại khởi xướng bởi sinh viên. Các DHClass-Hub là công khai có sẵn cho công chúng, duy trì liên tục trong học kỳ, và được chia sẻ bởi hai khóa học, làm cho lưu trữ của vấn đề mở và đóng cửa một nguồn tài nguyên có thể tìm kiếm ngày càng có lợi theo thời gian cho sinh viên và cho cộng đồng GitHub rộng hơn. Sự phát triển của kho là sáng kiến của một học sinh đã trở thành một trợ lý giảng dạy trong khóa học và viết hướng dẫn đi kèm của nó đối với việc tiếp cận GitHub tại dòng lệnh. Các tác giả thấy rằng những lợi ích quan trọng nhất để sử dụng GitHub không phải là để hỗ trợ những ưu tiên ngắn hạn của một khóa học học kỳ kéo dài, nhưng, đúng hơn, khuyến khích phát triển kỹ thuật số bền vững và cũng như các tài liệu, cả hai dự án của sinh viên và quá trình riêng của mình.

## Giới thiệu: Các GitHub Challenge

“Ngay từ đầu, tuy nhiên, ý tưởng đã được bắt đầu bằng cách estranging sinh viên đến từ các nghĩa thông thường của những gì một máy tính là gì và những gì sắp xếp của giao diện nó có,” quan sát Stephen Ramsay, giải thích lý do tại sao anh ta bắt đầu một khóa học tại lập trình cho nhân văn kỹ thuật số với định hướng mở rộng cho UNIX và giao diện dòng lệnh (Ramsay [2012,](https://link-springer-com.ezproxy.library.ubc.ca/chapter/10.1007/978-3-319-54226-3_15#CR8)mệnh. 13). Sự ra đời của sinh viên chưa qua đào tạo trong khoa học máy tính đến một giao diện dòng lệnh, với tất cả các hiệp hội mang tính ẩn dụ củathư mục *cây* vàchế biến *nĩa* và *các chi*nhánh,có giá trị triết học về sư phạm Ramsay, và chúng tôi cũng vậy, đã tìm thấy nó một trải nghiệm hấp dẫn của sự ghẻ lạnh hiệu quả trong các ứng dụng của chúng ta về GitHub trong nhân văn số khóa học Pitt-Greensburg của.

Chúng tôi đã thực hiện GitHub trong hoạt động ngày-to-ngày của hai liên quan chặt chẽ lớp Digital Nhân văn: Mã hóa và kỹ thuật số lưu trữ và Mã hóa và Data Visualization, với đề cương môn công bố công khai tại [http://newtfire.org/dh/.](http://newtfire.org/dh/)Các khóa học được cung cấp trong mùa thu và mùa xuân học kỳ, được trang bị công nghệ chồng chéo, nhưng họ không được dạy theo một cách trình tự, có nghĩa là sinh viên mới có thể bắt đầu với một trong hai sự sụp đổ hay quá trình mùa xuân, và sinh viên trở về có cơ hội để cố vấn cho sinh viên mới và hỗ trợ việc giảng dạy khóa học. Nhấn mạnh của chúng tôi trong mỗi khóa học là trên mã hóa văn bản và khả năng của mình để nghiên cứu, chúng tôi giải thích trong tài liệu học tập của chúng tôi:

Điều này đôi khóa học là tất cả về làm những điều thú vị với các văn bản sử dụng máy tính và công nghệ kỹ thuật số. Chúng tôi dạy các khóa học với trọng tâm là làm việc với các văn bản như hiện vật của văn hóa nhân loại hình chủ yếu bằng lời nói và chữ-các hình thức ngôn ngữ “được viết” chuyển giao cho nhiều phương tiện truyền thông (bao gồm cả hình ảnh và âm thanh) mà chúng ta có thể nghiên cứu với các công cụ máy tính mà chúng tôi thiết kế cho bản thân dựa trên những câu hỏi chúng tôi yêu cầu. Chúng tôi làm việc với các máy tính trong khóa học này là dụng cụ chính xác giúp chúng ta đọc và xử lý một lượng lớn thông tin, và điều đó dẫn chúng ta để làm cho các kết nối có ý nghĩa, yêu cầu loại mới của câu hỏi, và xây dựng các mô hình và giao diện để thay đổi cách đọc của chúng tôi và kinh nghiệm suy nghĩ như người tò mò về lịch sử loài người, văn hóa và sáng tạo (Beshero-Bondar 2016).

Trình tự hai học kỳ của các khóa học tại Pitt-Greensburg thu hút khoảng mười sinh viên mỗi học kỳ từ nhiều lĩnh vực chủ yếu là bên ngoài của khoa học máy tính, bao gồm cả Nhân học, Lịch sử, Văn học Anh, truyền thông, và Viết sáng tạo, với các sinh viên thỉnh thoảng trong Hệ thống Thông tin người sớm nhận ra rằng cách chúng ta nhấn mạnh việc xây dựng kiến trúc web với lập trình máy tính là một trải nghiệm ứng dụng mà họ đã bỏ lỡ trong môn học theo yêu cầu của họ. Trong số những người dân chủ yếu của sinh viên tham dự khoá học (chuyên ngành tiếng Anh), hầu hết chưa bao giờ kịch bản mã trước và đã hiếm khi mạo hiểm ra khỏi giao diện WYSIWYG quen thuộc trong các hệ điều hành Windows và Mac. Đa số những sinh viên này là dự đoán không quen với giao thức quản lý tập tin cơ bản (chẳng hạn như sử dụng phù hợp các phần mở rộng tập tin), hãy cho mình một môi trường shell. Chúng tôi hướng dẫn và trở về sinh viên cố vấn chia sẻ với các sinh viên mới kinh nghiệm của chúng tôi với XML và gia đình của các ngôn ngữ với một mục tiêu chính của việc áp dụng những công cụ như tính toán để điều tra câu hỏi nghiên cứu thông qua mã hóa và xử lý văn bản văn hóa. Trong quá trình học tập theo cách của họ xung quanh viết XSLT và XQuery sang HTML sản lượng và SVG, các sinh viên có được kinh nghiệm trong việc thiết kế các dự án mà trả lời câu hỏi nghiên cứu lựa chọn của họ, chẳng hạn như: làm thế nào chúng ta có thể thiết kế một giao diện web để giúp chúng tôi nghiên cứu các văn bản biến thể của một bài thơ, và bao nhiêu là một tập hợp các bài thơ thay đổi bởi các biên tập viên[(http://dickinson16.newtfire.org),](http://dickinson16.newtfire.org/)hoặc những gì chúng ta có thể tìm hiểu về các giả định về quyền sở hữu và quyền lực trong ngôn ngữ của phụ nữ công nhân công nghiệp và ông chủ của họ vào cuối thế kỷ XIX[(http://nelson.newtfire.org).](http://nelson.newtfire.org/)Để thiết kế các dự án như thế này, sinh viên của chúng tôi làm việc theo nhóm để lên kế hoạch và thực hiện một hệ thống mã hóa văn bản và sơ đồ quy tắc để phối hợp nỗ lực của họ, và họ không chỉ mã hóa văn bản, mà còn viết các chương trình để chuyển đổi mã của họ vào giao diện đọc sách xây dựng cho công chúng web. Họ mã để trích xuất thông tin và tóm tắt nó trong biểu đồ và đồ thị để hỗ trợ các “đọc xa xôi” của thông tin khó khăn cho con người để xử lý mà không cần hỗ trợ tính toán. Cụm từ “đọc xa” như đã giới thiệu bởi Franco Moretti liên quan đến việc áp dụng xử lý tính toán đến số lượng lớn các văn bản cùng một lúc để khám phá mô hình từ xa, và là một thay thế cho một sự nhấn mạnh truyền thống trong văn học và nhân văn về “đọc gần” của cá nhân văn bản (Moretti 2013).

Những dự án lớn và phức tạp đòi hỏi tinh thần đồng đội phù hợp với văn hóa của sự hợp tác trong khoa học nhân văn kỹ thuật số, nhưng họ rất khó để nhận ra và quản lý trong quá trình 15 tuần học kỳ bận rộn. Với các khóa học khác, công ăn việc làm, và nhiều hoạt động khác cạnh tranh cho sự chú ý của học sinh, làm việc theo nhóm dự án thỉnh thoảng là một khuấy trộn và quá trình bị gián đoạn đặc trưng bởi những nỗ lực đứt quãng trong 12-13 tuần đầu tiên sau đó là một vụ nổ của hoạt động cường độ cao trong những tuần kết luận như sinh viên áp dụng các loại khác nhau của mã hóa mà họ đã học và làm việc trên xuất bản của họ giao diện web, dữ liệu, phân tích và kết luận. Biết được điều này sẽ là một trải nghiệm khó khăn để quản lý một cách hiệu quả, trong những năm đầu tiên mà chúng tôi chạy các khóa mã hóa chúng tôi chỉ sử dụng các chức năng kéo-và-thả giao diện thuận tiện nhất và quen thuộc cho việc chia sẻ tập tin rằng hầu hết các sinh viên của chúng tôi đã biết và truy cập: Dropbox ( cung cấp miễn phí cho sinh viên) và Hộp (mà trường đại học của chúng tôi đang sở hữu một thuê bao, làm cho nó miễn phí cho học sinh sử dụng). Những giao diện này cung cấp một phương tiện hạn chế về kiểm soát phiên bản của hồ sơ dự án, có lẽ đủ tốt cho việc quảng cáo hoc trên một dự án tạm thời, nhưng không hệ thống lý tưởng cho việc theo dõi lịch sử phiên bản hoặc đối phó với chỉnh sửa đồng thời.

Chúng tôi đã đủ quen thuộc từ các dự án riêng của chúng tôi với khó khăn GitHub của, lạ của mình cho người lãnh đạo, phải được quan tâm về đòi hỏi sinh viên của chúng tôi để sử dụng nó. Tuy nhiên, kế hoạch của mỗi học kỳ mới nêu lên câu hỏi về việc liệu thử nó, và thực sự cố gắng nó một cách chính xác để mở rộng phạm vi của hướng dẫn trình của chúng tôi trong việc quản lý tập tin lưu tâm và để cung cấp cho sinh viên những kỹ năng phát triển thực tiễn có khả năng dẫn đến việc làm. Đồng nghiệp của chúng tôi, David J. Birnbaum, người dạy một Nhân văn kỹ thuật số tương tự như quá trình mã hóa tại cơ sở Pittsburgh láng giềng của Đại học Pittsburgh (xem [http://dh.obdurodon.org),](http://dh.obdurodon.org/)bắt đầu đòi hỏi GitHub cho phát triển đội dự án sinh viên của ông trong năm 2014 và giới thiệu GitHub với giao diện người dùng đồ họa (GUI) khách hàng máy tính để bàn, trong đó đã cho chúng tôi một số ưu đãi và tiền lệ cho việc kết hợp GitHub trong khóa học của chúng ta. Tuy nhiên, trong sau dự án sinh viên Birnbaum, chúng tôi phát hiện ra rằng thực hiện giới thiệu này là cuộc khủng hoảng dễ xảy ra khi học sinh thất bại trong việc đồng bộ hóa kho của họ và thấy giao diện khách hàng của họ là vô ích trong tranh với kết quả hợp nhất các cuộc xung đột, mà cần phải được giải quyết tại cửa sổ dòng lệnh bằng cách một giảng viên khóa học trong một cách ad-hoc.

Nhận thức được vấn đề như vậy và vẫn đánh giá cao sự phát triển kho của sinh viên và những kỹ năng họ đã học trong khóa học Birnbaum, chúng tôi vẫn không chắc chắn về việc liệu và làm thế nào để kết hợp GitHub trong Nhân văn số khóa học tại các tuần trong khuôn viên trường Greensburg trước khi bắt đầu học kỳ mùa thu trong Tháng Tám năm 2015. Beshero-Bondar cuối cùng đã đồng ý để thực hiện GitHub vào khóa học tại sự khăng khăng của Rebecca J. Parker, trợ lý nghiên cứu đại học của mình. Parker đã là trường hợp mà sinh viên của chúng tôi sẽ tốt nhất học thực tiễn quản lý tập tin lưu tâm và có được cái nhìn rộng lớn hơn của các dự án kỹ thuật số thú vị bằng cách tham gia các mạng xã hội nào GitHub, và cô lập luận thuyết phục nhất là dân lớp của chúng tôi hiện không có người truy cập khác với các khóa học mà có thể hướng họ đến GitHub. Mục tiêu chính của chúng tôi là yêu cầu sinh viên cộng tác với nhau trong các dự án cuối cùng của họ sử dụng GitHub, nhưng chúng tôi nhận ra rằng trừ khi chúng tôi thực hiện những kỳ vọng nguy cơ thấp để làm việc với GitHub trong một vị trí trung tâm mỗi ngày từ đầu khóa học, sinh viên bận rộn của chúng tôi có thể là quá thiếu kiên nhẫn để học hỏi khả năng của mình khi làm việc trên các dự án của họ trong các nhóm nhỏ tách biệt với hoạt động lớp hàng ngày. Thí nghiệm GitHub là nhiều khả năng thành công nếu sinh viên có thể quen làm việc gì mình vào việc duy trì các kho địa phương của họ vào đầu học kỳ, trước khi bắt đầu giai đoạn căng thẳng của hoạt động vào cuối của khóa học về dự án học kỳ của họ.

## Cơ Garage: Học Git và GitHub

giải pháp của chúng tôi là kết hợp sử dụng GitHub thông thường thành những kinh nghiệm hàng ngày của các hoạt động lớp bằng cách tạo ra một kho lưu trữ mà chúng ta đặt tên cho DHClass-Hub, tọa lạc tại <https://github.com/ebeshero/DHClass-Hub>. Mục đích của kho là một mục tiêu chung cho sinh viên và giáo viên hướng dẫn để làm việc cùng nhau về các hoạt động học tập của lớp. Kho không được thiết kế để tiếp nhận đệ trình cá nhân chính thức của bài tập, nhưng thay vì làm cho sinh viên để tìm kiếm sự giúp đỡ từ các lớp như một cộng đồng với bài tập về nhà và dự án mã hóa của họ phát triển, để chia sẻ các file có vấn đề với nhau để được giúp đỡ với gỡ lỗi, cũng như, đơn giản, định hướng đến môi trường GitHub trong một bối cảnh hỗ trợ lẫn nhau. Chúng tôi sử dụng GitHub sẽ chứng minh làm thế nào một lớp có thể làm việc cùng nhau như một đội bóng, giống như các nhà phát triển làm. Nó được thiết kế để có kinh nghiệm đầu tiên của học sinh với GitHub trước khi tạo các kho riêng của họ cho dự án mà họ quản lý một mình.

Sinh viên của chúng tôi cần thiết để đạt được sự quen thuộc với GitHub như một công cụ mạng xã hội được xây dựng trên hệ thống kiểm soát phiên bản Git. Git là một *hệ thống kiểm soát phiên*bản,hoặc một hệ thống được thiết kế để quản lý thay đổi các tập tin bằng cách lưu trữ một ký ức của mỗi hình thức mà họ đã chụp. Một số hệ thống kiểm soát phiên bản làm điều này trong một cách tập trung, bằng cách giữ một bậc thầy bản sao của một tập tin và lưu trữ tất cả lịch sử thay đổi của nó ở một vị trí duy nhất. Nếu hai người dùng chia sẻ một thư mục tập tin đó theo dõi lịch sử phiên bản một cách tập trung, và mỗi cố gắng để thay đổi cùng một tập tin và lưu thay đổi với vị trí cùng một lúc, các hệ thống tập trung tạo ra một bản sao của một trong các tập tin với một tập tin thay đổi tên, và các chủ sở hữu của thư mục sau đó phải quyết định phiên bản nào để giữ hoặc làm thế nào tốt nhất để dung hòa hai phiên bản. Git không hoạt động theo cách này bởi vì nó là một *phân phối hoặc phân* hệ thống kiểm soát phiên bảncấp,một hệ thống cho phép nhiều địa điểm lưu trữ để được điều trị tương đương. Những người cộng tác trên một dự án có thể sao chép và đóng góp các file của họ vào một không gian lưu trữ dựa trên web hoặc một *nguồn gốc từ* kho lưu trữxa,nhưng khi họ làm việc với các tập tin từ xa họ tạo ra bản sao của chúng trên máy tính của họ ở một vị trí được biết đến như một *kho lưu trữ tổng thể*, và Git quản lý thông tin về lịch sử phiên bản địa phương. Như tập thể dục định hướng đầu tiên của họ, sinh viên của chúng tôi ban đầu học cách làm việc với GitHub trong DHClass-Hub của chúng tôi bằng cách đầu tiên thực hiện một bản sao cục bộ của một kho lưu trữ nguồn gốc (gọi là *nhân bản* kho). Sau đó, họ hoặc là giới thiệu một tập tin mới vào kho lưu trữ địa phương của họ hoặc thay đổi một tập tin hiện có, và họ cập nhật kho từ xa bằng cách *cam kết* thay đổi của họ, trong đó bao gồm việc viết một thông điệp ngắn gọn mô tả sự thay đổi họ đang giới thiệu và thiết lập nó trong tài liệu kho lưu trữ từ xa của lịch sử phiên bản của nó. Cuối cùng, họ *đẩy* những thay đổi của họ (mà sẽ gửi những thay đổi trực tiếp đến kho lưu trữ từ xa), có nghĩa là *nếu* chủ sở hữu kho từ xa đã cấp quyền truy cập để chỉnh sửa các file trong thư mục từ xa, như chúng ta đã làm trong việc giới thiệu sinh viên của chúng tôi để các DHClass-Hub. Mà không được phép như vậy, người dùng sẽ phát hành một *yêu cầu kéo* cho chủ sở hữu của các kho lưu trữ từ xa, đó là một yêu cầu đối với các chủ sở hữu để *kéo* hoặcxem các thay đổi đầu tiên trước khi chấp nhận sự thay đổi mới cam kết trong lịch sử phiên bản của điều khiển từ xa. Đây là một thực tế phổ biến giữa các nhà phát triển mã nguồn mở và giúp họ làm việc như một cộng đồng mở rộng với bất cứ ai có ý tưởng tốt và lợi ích chung, và vì lý do này, các mạng phân phối phiên bản đôi khi được mô tả như một *trang web tin*cậy,hoặc một mạng lưới phân phối trong đó tập tin có thể được phát triển trong nhiều địa điểm và chia sẻ dựa trên thành tích của họ, hoặc ngược lại, từ chối hoặc từ chối khi họ gây ra vấn đề.

Chúng tôi đã cho tất cả các sinh viên của chúng tôi *đẩy truy cập* vào kho DHClass-Hub, có nghĩa là có hiệu lực mà họ chia sẻ trong quyền sở hữu của kho lưu trữ nguồn gốc từ xa. Trong một “cuộc sống thực” môi trường phát triển này có thể khó khăn để quản lý, nhưng với các giảng viên và sinh viên làm việc cùng nhau chia sẻ một mục tiêu chung, các giảng viên đứng bên để *trở lại* (hoặc undo) cam kết kho gây thiệt hại (như trong xóa thư mục tập tin chúng ta cần), và chúng tôi xử lý các DHClass-Hub là một môi trường thực hành anyway. Không có gì có thể xảy ra ở đây sẽ là thảm họa cho các hoạt động của chúng tôi bởi vì chúng tôi đang chuẩn bị để khôi phục lại bất cứ điều gì mà dường như đã bị mất. Trong vòng hai tuần đầu tiên của lớp chúng tôi, học sinh được hướng dẫn từng đầu tiên và quan trọng nhất để *đồng bộ hóa* các kho địa phương nhân bản của họ với điều khiển từ xa DHClass-Hub khi họ bắt đầu làm việc tại máy tính của họ. Này bao gồm việc định kỳ kết hợp thay đổi từ kho lưu trữ nguồn gốc từ xa để mang lại các kho địa phương của họ được cập nhật. Sau đó, họ đang từng hỏi để giới thiệu một sự thay đổi hoặc một tập tin mới, cam kết thay đổi, và đẩy nó để điều khiển từ xa. Cuộc diễn tập đầu tiên thường liên quan đến sinh viên đẩy những điều mà họ muốn chia sẻ với các lớp học, chẳng hạn như một bức ảnh từ một chuyến đi, và khi họ kéo vào những thay đổi của nhau họ thường có được kinh nghiệm đầu tiên của họ khi làm việc với một hệ thống kiểm soát phiên bản phân phối và nó web of Trust.

GitHub, như khác biệt với Git, là một dịch vụ lưu trữ web cho kho từ xa và kết hợp chúng với các công cụ mạng xã hội tăng cường hợp tác. GitHub kho trên web có thể lưu trữ wiki và một hội đồng quản trị các vấn đề có thể trợ giúp tài liệu quy định dự án của một nhóm nghiên cứu và quy trình làm việc. Hội đồng quản trị các vấn đề cung cấp đóng góp và du khách với chiếm GitHub cơ hội để gửi hoặc *mở* một vấn đề cần đặt câu hỏi hoặc tham gia vào một cuộc thảo luận ren của một vấn đề. Mỗi vấn đề mới được gán một số như một hệ thống vé giúp đỡ, và như các vấn đề được giải quyết một cộng tác viên đánh dấu chúng như *khép*kín.Vấn đề khép kín vẫn có thể tìm bên cạnh những người mở nhưng được ẩn từ xem trên trang web của kho. Đóng góp kho có thể tìm kiếm thông qua tất cả các vấn đề mở và đóng cửa cho trí tuệ tập thể của kho lưu trữ trên một chủ đề cụ thể, và họ có thể mở lại các vấn đề đóng khi cần thiết, hoặc tham khảo chúng với một liên kết tích cực trong các cuộc thảo luận về các vấn đề khác. Cộng tác viên cũng có thể đăng các liên kết đến các tập tin trong kho GitHub từ xa hoặc đến các trang ghi lại tất cả những thay đổi của một cụ thể cam kết trong lịch sử kho từ xa. GitHub thậm chí có phương tiện thêm ý kiến liên quan đến đường dây đặc biệt của mã trong một cam kết, và là một cách để gửi tin nhắn trực tiếp cho các thành viên GitHub từ những ghi chú, một vấn đề, hoặc một wiki. Chức năng này làm cho GitHub đặc biệt tiện dụng cho các tài liệu chính xác và tinh chỉnh trên các tập tin đã cam kết.

Các DHClass Hub tổ chức khối mẫu mã mà các giảng viên muốn chia sẻ từ lớp, cùng với các trang wiki giữ hướng dẫn chung, và một hội đồng quản trị các vấn đề rất tích cực nơi học sinh gửi câu hỏi và mẫu mã họ đang làm việc. Trong khi giáo viên hướng dẫn gửi mẫu mã để các thư mục mà các sinh viên có thể kéo trong để nghiên cứu, các kho lưu trữ cũng chứa một thư mục quan trọng, chúng tôi đã đặt tên “Gỡ rối”, nơi sinh viên được mời đẩy mã vấn đề của họ khi gọi để được giúp đỡ với gỡ lỗi từ cộng đồng lớp. Với thư mục Troubleshooting và các vấn đề hội đồng quản trị, các chức năng DHClass-Hub như một gara hợp tác ở đâu, là một trong những sinh viên của tôi đã mô tả nó, họ có thể đối xử với các vấn đề với mã của họ tất cả cùng nhau như “sự xoay lên xe của bạn vào gara để làm cho nó cố định và có các sinh viên và giáo viên hướng dẫn trực tuyến khác để giúp sửa chữa nó.”Ẩn dụ gara dường như kêu gọi hiệp hội cụ thể của nó với các công cụ mà sửa chữa máy móc, và trong không gian nhà để xe này, sinh viên có được kinh nghiệm bằng việc cả khách hàng và các cơ chế, áp dụng các công cụ của họ để giúp đang của nhau chạy trơn tru.

Chúng tôi phát hiện ra rằng lớp chúng tôi có chức năng một cách cải thiện với DHClass-Hub. Trong đó môi trường ảo, chúng ta trải qua sự gắn kết cộng đồng lớn hơn chúng ta có thể đạt được một trong hai trong lớp hoặc với một diễn đàn thảo luận trực tuyến thông thường (như chúng ta đã thực hiện trong những dòng máy trước tất nhiên chúng tôi). Sự kết hợp giữa thảo luận theo luồng kết hợp với khả năng đẩy và kéo tập tin và để áp dụng và xem xét dòng bình luận trên mã mang lớp như một toàn thể tập thể đến gần với công việc của họ với nhau theo một cách ủng hộ. Chúng tôi trải qua mình là một cộng đồng làm việc cùng nhau một cách hệ thống về việc sửa chữa bằng cách làm lỗi thử nghiệm và tài liệu có thể sửa theo hình thức bình luận trong markdown trên một mã khác của. Thông thường sinh viên sẽ cố vấn và hướng dẫn người khác khi giáo viên hướng dẫn đã có sẵn, và nhiều lần sinh viên tìm thấy cách bất thường để giải quyết vấn đề hoặc đưa ra (và trả lời) câu hỏi mà các giảng viên đã không lường trước được.

Như vậy gara GitHub của chúng tôi bất ngờ trở thành các locus của một trải nghiệm hiệu quả lật học tập, trong đó các sinh viên chịu trách nhiệm về quá trình học tập của họ và giáo viên hướng dẫn học cùng với chúng. Theo định nghĩa xác nhận bởi mạng Learning Flipped và bằng sáng chế học tập lật Jonathan Bergmann và Aaron Sams, “lộn học là một phương pháp sư phạm, trong đó hướng dẫn trực tiếp di chuyển từ không gian học tập nhóm để không gian học tập cá nhân, và không gian nhóm kết quả là chuyển đổi thành một môi trường học tập tương tác năng động, nơi các nhà giáo dục hướng dẫn sinh viên khi họ áp dụng các khái niệm và tham gia một cách sáng tạo trong vấn đề”(Bergmann và Sams [2014).](https://link-springer-com.ezproxy.library.ubc.ca/chapter/10.1007/978-3-319-54226-3_15#CR1)Chúng tôi đã hy vọng giáo viên hướng dẫn đơn giản chỉ để quen làm việc gì sinh viên của chúng tôi đến một công nghệ xa lạ, nhưng chúng tôi đã thành công bất ngờ trong việc tạo ra một môi trường cho sinh viên của chúng tôi để chịu trách nhiệm về quá trình học tập của họ và tham gia vào với nhau trong một quá trình peer-học.

Phần lớn những gì đã xảy ra trong khóa học của chúng tôi trong năm vừa qua tương tự như thí nghiệm học tập lật thành công trong khóa học khoa học máy tính như văn bản, ví dụ, bởi Michael Skerpan và Tom Yeh, có khóa học trong “Thành viên làm trung tâm Thiết kế và phát triển” tại Đại học Colorado, Boulder tiến hành như dự án phát triển với các sinh viên làm việc trong hợp tác “hack-một-thons” và giáo viên hướng dẫn cung cấp “tùy chỉnh thông tin phản hồi on-the-spot” để hướng dẫn sự tiến bộ của họ (Skirpan và Yeh [2015).](https://link-springer-com.ezproxy.library.ubc.ca/chapter/10.1007/978-3-319-54226-3_15#CR9)Nổi bật hơn, kinh nghiệm của chúng tôi giống như một trường hợp nghiên cứu bởi Csaba-Zoltán Kertész, người sử dụng GitHub như một chiến lược để tập trung làm của học sinh trong việc phát triển phần mềm cộng tác. Kertész thực hiện GitHub như một phương pháp giảng dạy nhằm khuyến khích sinh viên tại Khoa Điện tử và Máy tính tại Đại học Transylvania để tương tác với nhau để phát huy tối đa “những tư duy phê phán và tinh thần làm việc theo nhóm” (Kertész [2015).](https://link-springer-com.ezproxy.library.ubc.ca/chapter/10.1007/978-3-319-54226-3_15#CR3)Các mặt đối mặt lớp sống của Nhân văn kỹ thuật số của chúng tôi dĩ nhiên có lẽ không nhìn như giống như một “lật lớp học” như những người đại diện trong các nghiên cứu trường hợp ở trên, kể từ khi chúng tôi bảo lưu thời gian lớp trong người để thảo luận về tất cả cùng nhau một công nghệ mới hay xem lại bài tập và giới thiệu các phương pháp mới, nhưng chúng tôi vẫn thấy rằng sinh viên đã đạt được sự thoải mái trong việc nói lên và đặt câu hỏi trong lớp, và dường như chuyển giao kinh nghiệm peer-learning của họ từ ảo DHClass-Hub của chúng tôi để các lớp vật lý. Môi trường học tập kiểm soát nhiều hơn trong lớp học vật lý của chúng tôi có thể hữu ích để tổng hợp “takeaway”, hoặc những gì điều quan trọng để học hỏi từ kinh nghiệm của sinh viên trong môi trường ảo lộn, và để chuẩn bị cho giai đoạn tiếp theo của hoạt động trong khóa học.[1](https://link-springer-com.ezproxy.library.ubc.ca/chapter/10.1007/978-3-319-54226-3_15#Fn1) Tính năng động nhóm lớp của chúng tôi là rõ ràng nhất và sống động trên các trang Vấn đề của DHClass-Hub của chúng tôi, nhưng trong lớp học vật lý, chúng ta thấy rằng lợi ích đáp ứng tất cả cùng nhau ở một nơi là cơ hội để đồng bộ hóa và xem xét kinh nghiệm cá nhân trước khi tiến hành trong cuộc thảo luận trực tuyến không đồng bộ và thực hành thí nghiệm với mã.

Các DHClass-Hub tạo điều kiện làm việc một mình trên thời gian của chính mình như sinh viên kéo tập tin lớp học để xem trong máy nhái địa phương của họ về kho lưu trữ, và cộng tác trong việc phát triển không gian giảng dạy như sinh viên gửi tập tin và các vấn đề tài liệu riêng của họ. Trong DHClass-Hub, các hoạt động sau xảy ra trên một cơ sở hàng ngày trong một học kỳ làm việc:

* Giảng viên đẩy mẫu mã theo mô hình trong lớp, do đó sinh viên có thể kéo trong những tập tin và truy cập được địa phương rà soát về thời gian riêng của họ.
* Học sinh gửi câu hỏi và các vấn đề với bài tập về nhà của họ mã hóa bằng cách mở các vấn đề như sự giúp đỡ vé. Họ viết vé trong markdown, và giao diện web GitHub trở thành một cơ chế để suy nghĩ theo cách của họ thông qua một vấn đề ngay cả trong quá trình kêu gọi sự giúp đỡ. Các sinh viên tham gia vào các cuộc nói chuyện với nhau để gỡ lỗi mã của họ và so sánh vấn đề họ đang gặp phải, và họ đóng những vấn đề riêng của họ khi họ được giải quyết.
* Học sinh đẩy mã của họ vào một thư mục Troubleshooting, và yêu cầu giúp đỡ với lời bình luận dòng chỉnh sửa để xác định dòng có vấn đề và đề xuất sửa lỗi.

Chúng tôi sử dụng các kho cung cấp một lời nhắc nhở hàng ngày và tăng cường mã mà không phải là điều chúng tôi mong muốn được hoàn hảo, và rằng trong các nhà phát triển cuộc sống thực trợ giúp lẫn nhau và có được kinh nghiệm khi làm như vậy. Các tăng kinh nghiệm phát triển cá nhân thông qua đối phó với những người khác trên trang DHClass-Hub Các vấn đề được xây dựng vào yêu cầu thảo luận về khóa học, nhưng nhiều hơn so với một yêu cầu, nó là một phương tiện chính mà sinh viên đi từ nghe tiếp thu và xem giáo sư để học tập tích cực. Ngoài thảo luận unmediated, việc sử dụng GitHub đã trở thành một kinh nghiệm thường xuyên viết về kỹ thuật, kết nối việc thực hành mã hóa với tài liệu hướng dẫn tăng cường bởi giao diện markdown GitHub. Trang Các vấn đề đóng vai trò như một giàn giáo trung gian giữa văn bản trong các cụm từ quen thuộc và câu và cú pháp ít quen thuộc của các mã được bực bội một sinh viên tại bất kỳ thời điểm nào. Ban đầu sinh viên đôi khi sẽ gọi để được giúp đỡ mà không giải thích bản thân và không cung cấp mẫu có liên quan từ mã của họ, nhưng dần dần họ đã hướng dẫn để tăng độ đặc hiệu thông qua thông tin phản hồi từ nhóm.

Chúng tôi tính năng ví dụ ở đây đối thoại từ hội đồng quản trị DHClass-Hub vấn đề để chứng minh các loại văn bản cho sinh viên tham gia vào. Việc trao đổi ngắn gọn dưới đây, có thể xem từ [https://github.com/ebeshero/DHClass-Hub/issues/26,](https://github.com/ebeshero/DHClass-Hub/issues/26)gắn liền một bài tập về nhà trên áp dụng biểu thức thông thường để up-chuyển đổi một phiên bản đồng bằng văn bản củaBernard Shaw *Pygmalion* đểXML. Một trong những thách thức là để xác định vị trí một mô hình mà có thể giúp cô lập tất cả các hướng sân khấu trong vở kịch.

CodyKarch: Này, tôi đang cố gắng để gắn thẻ các dấu ngoặc vuông, nhưng tất cả những gì tôi đã cố gắng không hoạt động. Bất kỳ gợi ý hoặc đề nghị?

ghbondar: bạn có thể tìm thấy những dấu ngoặc vuông ok trong văn bản sử dụng regex? \ [Và \] dường như làm việc.

nlottig94: Tôi sử dụng \ [. +? \] Để tìm tất cả các hướng giai đoạn trong tài liệu! Oh, và tôi đã “chấm phù hợp với tất cả” kiểm tra! Hi vọng điêu nay co ich!

CodyKarch: Wow ... Xin lỗi, tôi giữ cố gắng đó, và nó sẽ không làm việc. Bây giờ tôi nhận ra rằng tôi cần thiết để biến Dot Trận Tất cả về. Lỗi lầm ngớ ngẩn. Cảm ơn.

CodyKarch đóng này vào ngày 4 tháng 10 năm 2015

Dưới đây chúng ta có thể nhìn thấy một sinh viên yêu cầu giúp đỡ một cách tổng quát, và được nhắc nhở với biểu thức thông thường cụ thể để cố gắng, nhưng các câu trả lời khiến ông nhận ra lỗi của mình một cách độc lập (ông cần thiết để thiết lập một tính năng mà kiểm soát việc giải thích các dấu chấm trong <oxy /> trình soạn thảo XML của chúng tôi). Ở đây, ghbondar là một giảng viên khóa học và nlottig94 là một sinh viên, vì vậy chúng ta thấy một ví dụ tốt về một sinh viên nhảy vào để hỗ trợ một người đang bị mắc kẹt.

Trong một vấn đề khác đăng tải sau này trong quá trình[(https://github.com/ebeshero/DHClass-Hub/issues/43)](https://github.com/ebeshero/DHClass-Hub/issues/43)là sinh viên đã được học tập để mã XSLT, chúng ta có thể nhìn thấy chúng sử dụng thư mục Troubleshooting của chúng tôi để đẩy mã có vấn đề khi hỏi để được giúp đỡ. Các sinh viên khác chuyển đến rất nhanh chóng để giúp trước khi các giáo sư đã có thể trả lời. Học sinh thể hiện sự hiểu biết của họ với GitHub ở đây như là một cách để giúp giải quyết một trong những vấn đề của người khác và bình luận trực tiếp trên mã của một file khác bằng cách để lại tin nhắn trong hội đồng quản trị các vấn đề. Lưu loát của họ với GitHub và cơ cấu hỗ trợ hợp tác của họ được thể hiện rõ ở đây:

blawrence719: Tôi đang cố gắng làm việc giao XSLT thứ hai, nhưng tôi không nhận được bất cứ điều gì trong cửa sổ đầu ra của tôi trừ h1 tôi. Tôi cố gắng để theo cùng từ trong ví dụ lớp như chặt chẽ nhất có thể trong khi gắn bó với việc chuyển nhượng trong đó liệt kê được lệnh hoặc có thứ tự, vì vậy tôi không chắc chắn chính xác nơi tôi đã đi sai. Tôi đẩy tập tin của tôi vào thư mục gỡ rối. Tôi đã có thể hiểu được những gì đang xảy ra trong lớp học, nhưng tôi hoàn toàn bị mất trên của riêng tôi. Tôi không chắc chắn khi tôi đang nghĩa vụ phải được phù hợp với sự vật và chính xác những gì tôi nên phù hợp và tôi không hiểu làm thế nào để làm cho họ ra thành một danh sách.

nlottig94: Được rồi, tôi đưa một ví dụ lập theo các ví dụ lớp trước đó. Đó là lottig\_10-20\_extraPractice. Nó có thể giúp đỡ một số! @ blawrence719

rjp43: @ blawrence719 Tôi cũng đẩy các file của tiến sĩ B. từ lớp ngày hôm nay để tham khảo. Hãy nghĩ rất đơn giản về những gì chúng ta muốn. Chúng tôi muốn có một danh sách tổ chức các loại của các tổ chức và bên trong mà chúng ta muốn danh sách chưa được tổ chức của tất cả các tên tổ chức. Tôi sẽ xem xét hồ sơ của bạn trong khắc phục sự cố bây giờ, nhưng tôi muốn đầu tiên làm cho nó rõ ràng chính xác những gì bạn nên tìm kiếm.

rjp43: Ok như vậy từ những gì tôi nhìn thấy bạn đang so khớp, nhưng bạn không được chọn. Bên trong mục danh sách của bạn, bạn cần phải xác định chính xác những gì bạn muốn XSLT để lấy! @ blawrence719 tôi đã thực hiện một số ý kiến trong tập tin của bạn ... đi kiểm tra những người ra và mất một peek tại các ví dụ từ @ nlottig94 và @ebeshero. Hãy cho tôi biết nếu bạn nhận được nó!

blawrence719: Cảm ơn bạn @ nlottig94 và @ RJP43, cả hai bạn đã giúp rất nhiều, tôi nghĩ rằng tôi figured it out!

Trao đổi dài hơn chứng tỏ tinh thần hợp tác cùng tăng cường bởi GitHub sử dụng. Xem ví dụ việc trao đổi giữa brookestewart và spadafour vào “Schematron Câu hỏi” <https://github.com/ebeshero/DHClass-Hub/issues/70> như một cuộc trò chuyện cùng có lợi liên quan đến sinh viên gỡ lỗi mã của nhau. Học sinh tương tác với các giáo sư đôi khi dẫn đến những khám phá ngoạn mục cho cả hai, như trong Issue với tựa đề “Đếm biệt-giá trị trong vòng for-each” [https://github.com/ebeshero/DHClass-Hub/issues/60.](https://github.com/ebeshero/DHClass-Hub/issues/60)Ở đây nói chuyện của giáo sư với spadafour trên mã dự án cụ thể của ông đã dẫn cả hai để khám phá một phương pháp quen thuộc. Chúng tôi dự định viết một hướng dẫn mới cho quá trình dựa trên trao đổi và thông tin chúng tôi ghi nhận trong vấn đề GitHub này.

Biên bản chi tiết về vé giúp đỡ về trang DHClass-Hub vấn đề của chúng tôi đã để lại một dấu ấn lâu dài trên thực tế giảng dạy của chúng tôi. Câu hỏi, nhận xét và khám phá của học sinh đã dẫn đến sự cải thiện đáng kể trong hướng dẫn giảng dạy của chúng tôi và bài tập về nhà. Như các sinh viên đạt được kinh nghiệm với GitHub, tất cả đã trải qua một vài chúng tôi *xung đột*nhập,hoặc tình huống trong đó Git không thể vào những thay đổi quyết tâm riêng của mình được giới thiệu bởi nhiều người dùng. Như chúng ta đã học được cách để giải quyết những xung đột, chúng tôi phát hiện thêm bằng chứng về lợi ích của tài liệu cẩn thận GitHub, và các cuộc khủng hoảng tạm thời thúc đẩy một số các sinh viên hiểu rõ hơn về quy trình công việc Git và thậm chí để giúp đỡ với thử nghiệm trên của riêng mình để tạo ra những xung đột và giải quyết chúng. Một số sinh viên của chúng tôi giờ đây đã tung ra GitHub kho của riêng mình cho đề tài cấp cao của họ và các dự án khác mà không phụ thuộc vào các khóa học của chúng tôi mã hóa. Trên đầu trang của những lợi ích thực tế của điều khiển phiên bản, mức tăng trí tuệ của định hướng sinh viên vào quá trình tư duy có hệ thống GitHub đã là vô giá, vì có sự ghẻ lạnh hiệu quả tiết kiệm và chia sẻ tập tin trong một môi trường hợp tác mà chúng tôi tổ chức mình và các sinh viên của chúng tôi vào một nhóm phát triển. Việc thực hiện GitHub trong lớp học của chúng tôi đã hiệu defamiliarized mối quan hệ của sinh viên để máy tính của họ và để giáo viên hướng dẫn của họ, và đã cho sinh viên của chúng tôi cơ hội duy nhất để chịu trách nhiệm soạn thảo các tài liệu giảng dạy, trong đó có sinh viên tác giả của chúng tôi hướng dẫn về sử dụng Github tại [http: /](http://newtfire.org/dh/explainGitShell.html)/newtfire.org/dh/explainGitShell.html.

## Giao diện người dùng đồ họa hoặc Command Line?

Rõ ràng, việc sử dụng thường xuyên của học sinh của GitHub hưởng lợi ngày-to-ngày hoạt động của nhân văn kỹ thuật số của chúng tôi tất nhiên, nhưng việc họ sử dụng Git được giới hạn trong nhiệm vụ kéo những thay đổi và đẩy các file mới vào một kho lưu trữ từ xa duy nhất mà tất cả mọi người cùng nhau quản lý. Thực hiện của chúng ta về DHClass-Hub không yêu cầu học sinh đưa ra *các yêu cầu kéo* hoặc để tạotrưng riêng mình *dĩa* củakho, cả hai đều là tiêu chuẩn thực hiện trong cộng đồng GitHub. As Parker explains in her Github tutorial:

Part of the open-source etiquette of GitHub is that developers are encouraged to copy and then improve code originally created by others. GitHub supports the creation of derived projects through forking. When you fork a repo, you create a copy of it under your own account, where it acquires an independent identity. You can do anything with a repo you created by forking someone else's project that you can with a repo you created from scratch. The moment that you create the fork, your new repo is no longer synchronized with changes in the repo from which you copied it originally. If you fork a repo and make changes that you would then like to contribute to the original source, you can issue a pull request, inviting the developers of the original repo to merge your forked repo's changes into their original project (Parker [2016](https://link-springer-com.ezproxy.library.ubc.ca/chapter/10.1007/978-3-319-54226-3_15#CR6)).

Had we necessitated forking of the DHClass-Hub we would have introduced new complexities and changed the social dynamic of the experience. Instead of a unified experience of “we're all in this together,” each student would be working primarily in an isolated remote repository at a greater distance from the repository we instructors created. Students would need to work with Git in a more advanced way than is possible with the standard GUI Windows and Mac desktop client with which they were working at first (available from <https://desktop.github.com/>). They would need to learn how to work with Git at command line, an environment with which most of our students were not familiar, and given the intensity of our other course requirements we thought it best in fall 2015 not to demand this knowledge of them. We changed our minds by the start of spring 2016.

As we planned the course in fall 2015, we thought that our students should begin learning GitHub with the GUI client because creating a new repository is easier there than issuing *gitconfig* (or configuration) commands at the command line. The GUI client helps to make clear the relationship between the remote repository and one's cloned local repository, and whether or not these are in sync with each other. Furthermore, it provides a convenient space to view a repository from three possible access points: the command line, the local file explorer, and the remote origin. If needed in a crisis, students could open a command line shell and issue specialized commands, but we thought such a scenario unlikely or perhaps best handled by the instructors. We changed our perspective after crises erupted in November 2015, late in the fall semester. Students working in project teams in groups of four or five had created their own project repositories, and in these fully student-managed repository environments, they began to experience serious problems that had not surfaced in their use of the DHClass-Hub. When one student had neglected to sync his project repository regularly, had not pulled in updated files in three weeks, and pushed a new file he had created, the other team members were appalled to discover their recent files and directories overwritten, apparently gone, and we instructors worked together with the students, hunting through online tutorials on how to *revert* or *roll back* the repository to the point before the problematic commit, which required us to learn many new Git commands that could only be issued at command line. More frequently students encountered problems with *merge conflicts*, or situations in which a sync of the local with the remote repository cannot be completed because a file that a local user, say, Anne, has been editing was updated by another contributor, Bob, in the remote repository before Anne had a chance to commit her changes there. When Anne attempts to pull in changes from the repository, Git prevents her from doing so before she has *committed* or *stashed* her changes (that is, determined whether her changes should be shared with the remote repository or set aside, reserved from the remote). If she commits her own changes and proceeds to sync with the remote, Git might fail to *merge* the changes that Bob has made with those that Anne introduced. The GUI client led us to the command line to resolve these conflicts and to read messages guiding the resolution process. GitHub leaves a record of changes in the file in question and instructs the user to open the file, review all marked changes and select which ones to keep and which to discard. While reverting bad commits and resolving merge conflicts was stressful for students and instructors alike, it was also collectively educational for all of us in reinforcing the importance of regular syncing *before* starting work and in proving the safety of working with Git's distributed version control system.

Our experience of these crises in the fall 2015 semester left us wondering whether we ought to teach students to work with Git at command line from the beginning, bypassing the GUI client which sometimes seemed to function inconsistently. The team leader of the project repository we rescued by reverting the bad commit was convinced that the crisis might have been averted had she put the other team members at a greater distance, so that they could not push changes without permission from her as the repository owner, in other words, so that they would have to issue pull requests. We considered whether and how to teach students the command line skills they would need to have project team members work in their own forks of a project repository in order to ensure greater security of team projects and also to encourage students to learn more skills in working with an expanded range of Git commands. For the next semester's Digital Humanities class we decided to offer more training at the command line, and we found we needed to prepare our own tutorial to make this feasible. Since Git is used by many developers, documentation is both plentiful and diverse, depending on the implementation, and we quickly discovered that for each aspect of Git and GitHub that we wanted to implement in the course, we needed to refer to several external tutorials, many of which were written for professional developers who may be new to Git, but—unlike our students—not to version control or project management in general. In creating our own tutorial, Rebecca Parker's “Guide to Git Shell” (<http://newtfire.org/dh/explainGitShell.html>), we wanted to highlight the simplicity of using Git while demystifying, rather than avoiding, its numerous complex functions, and we also wanted the students to have a step-by-step roadmap in one place that introduced all of the specific aspects of Git and GitHub we hope they will learn in our course.

In spring 2016, we continued to introduce students to Git and GitHub with the desktop GUI client and with all students sharing equal ownership of the DHClass-Hub. However, we implemented a new workflow for student project teams, requiring each team member to create a fork of the team's repository. This required that students begin working with Git at command line and that one team leader would be in charge of the main repository, while everyone else in the group would work in a fork. Each team member's forked repository needed to be set to recognize their remote origin repository as *upstream*, or a remote origin to their own forked remote, in order to pull in updates from the other team members. In our now more carefully managed environment we wanted the *pushing* of files to the main upstream repository to require a screening process by the team leader, a situation neatly answered by the Git *pull request* model. Although forking and the use of pull requests prevents code errors by project members from contaminating the remote origin repository, the team leaders as owners of the project repositories still faced the risk of adding problematic code of their own through pushing directly in the old way. By creating development *branches* in the main repository, or alternative versions of a repository meant to be reserved from the master branch that participates in remote file sharing, and even by creating new forked repositories of their own, team members could experiment with development code without running the risk of contaminating the remote repository, which could remain in a stable state with files ready for sharing and use by all. GitHub's *fork and branch* workflow therefore seemed ideal for the operations of our student project teams, where we expected each team member to take responsibility for a particular aspect of a digital project's development usually reserved to a particular set of files necessary for project development. Ideally, an individual team member can work on a file in a development branch, and commit it to the master branch when the file is ready for others to work with in their project tasks.

Although the GUI simplifies the initial configuration process, which makes it a convenient starting point for students to begin to learn how Git works, the command line configuration is not impossible for beginners, it is well documented, and it puts more file management tools at their disposal. Once students have configured their accounts with the GUI and understand how to navigate to their repositories through the file explorer and command line, the primary advantage of the GUI ends, while much more can be accomplished at the command line. In practice, however, by the conclusion of spring 2016, most of our students had not gained facility or confidence with the command line functionality and asked for a great deal of help from the instructors each time they needed to commit changes to their project repositories. We were disappointed to find that our new system for managing projects did not run as smoothly as we had hoped, mainly because students preferred to work with the GUI client and required a lot of help with pushing and pulling from an upstream repository. This is understandable, since these actions are not as common or familiar as the simple, immediate Git workflow of the (unforked) DHClass-Hub, where everyone shares push access to the main repository.

We find ourselves, then, at a decisive turning point in our implementation of GitHub. As before, when deciding whether to require its use in the first place, we recognize that it is best to habituate students to a workflow on a daily basis. We do not want to lose the immediacy of the DHClass-Hub as a central, shared work environment to which everyone has push access, but if we want to habituate our students to working in forks and branches, we recognize that this, too, should not be restricted to project work. We have now come to think that the command line approach should be taught from the beginning of the course, precisely because our students have more to learn from exploring that environment as fully as possible. In developing our new command line GitHub tutorial, we discover that learning to issue pulls, commit messages, and pushes in the Git shell or command prompt makes the user aware of which particular files are being added and committed to their local directories and gives a clearer representation of any branches and remote connections of each repository than would be available in the GUI. While the GUI client does display files to be committed, it becomes more confusing than the command line when working with forks and branches. Echoing Stephen Ramsay, giving our students experience at command line from the start of the course may be the best method to learn through *estrangement* from the kinds of mediating interfaces to which they are accustomed.

As our Git and GitHub use increases from semester to semester, as we uncover more and more commands that can be used to solve and prevent issues, and as we introduce more of the features offered with Git, we will continue revising the command line tutorial to highlight the most useful functions for our student population. We are faced with a decision about how to improve our students' experience with GitHub at the start of fall 2016. One choice is to institute regular training throughout the course in the fork and branch workflow, mandating practice on a regular basis with regular weekly exercises that involve creating forks and branches and responding to pull requests. These assignments might encourage our community of instructors and students to become more comfortable with the command line and Git's distributed workflow. An extended series of homework exercises on GitHub Project Management , orchestrated once per week, might involve students setting up individual repositories dedicated to particular coding assignments or website development. We might ask each student to fork another student's repository and offer regular edits and comments on work there, so all students are better habituated to working as project managers on their own repositories and also as a team member operating in a fork of someone else's repository. These exercises would prepare students to work with greater confidence in their own ability to manage Git at the command line and with more independence from the instructors during the team project phase of the course, when we aim for them to work effectively together, keep each other informed, and take responsibility for keeping their forks and branches up to date on their own, as real developers do.

The other alternative is simpler and perhaps just as effective: Treat the project repositories in the same way as the DHClass-Hub, following our original model in fall 2015, *without*instituting the fork and branch workflow. However, train students to work with GitHub at command line from the beginning and emphasize regular monitoring of their repositories and syncing with their remote origin. Stand ready with tutorials and guides to assist students with merge conflicts and to revert or roll back problematic commits in case we have to resolve crises of the kind we experienced in November 2015. Students would only be required to gain facility with a small set of Git commands, enough to habituate them to GitHub as a workflow, and would still gain real development experience in the process. At the start of fall 2016, this second alternative seems easier for us to implement in balance with the many other topics and assignments we cover in our course.

## Not a Virtual Classroom but a Public Garage

As we developed our DHClass-Hub and its accompanying tutorials, we were aware of GitHub's own Education Program and Classroom Guide (see <https://education.github.com/>) and have debated whether to follow GitHub's preferred course organization method. GitHub's special program for educators automates the full construction of a course from assignment delivery to students to student homework submissions and development of semester projects, and it generates for educators a GitHub organization with a separate repository for each assignment, which may be public or private. From our point of view, the GitHub Classroom package replicates many features of other course management systems, such as Blackboard. We applaud the GitHub Classroom as an efficient all-in-one classroom management system, particularly in its use of GitHub's tools to track students' submissions and efforts over time. However, we find the Classroom Organization structure does not meet our use-case because it is too circumscribed to the delivery of content within a set framework and to set groups of students. What is prioritized are the needs of a course in its moment, separated from other semester offerings and requiring even separate repositories for each assignment, with the goal of managing and tracking teacher and student contributions within a narrow time frame. GitHub's Classroom Organization has little to offer the instructor who repeats a course over multiple semesters and wishes the repository to contain a searchable knowledge bank of shared resources that develops over a long period of time over multiple cycles of a course.

We found it easier to customize our use of GitHub to our specific needs by working within a single repository instead of adapting the GitHub Classroom model, which was built on top of specific assumptions and practices of pedagogy. The Classroom approach diverges from our main purpose, which is to train our (multiple) classes to work together as a combined community of developers. Our DHClass-Hub is the central locus of two coding courses we teach, and it houses the aspects of the course that are open to our community of coders and that extend into the public sphere. We instructors make clear from the beginning of our course that working in GitHub means working in public and not in the safe restricted confines of an academic course shell, and that student work in GitHub repositories is something that becomes a part of their public history on the internet. We encourage our students not to be fearful of this, but to embrace it as an opportunity to develop a documented history of their learning and their project development that they could easily pass on to future employers or graduate programs. While we wondered at first if students might find working in the public eye to be intimidating, so far we have discovered otherwise, that they embrace the opportunity to work in the public web. What appeals to us is the social nature of GitHub as a collaborative community that extends beyond the here-and-now of a particular repository's activities. We transform our GitHub repository at the end of each semester and at the beginning of each new one, removing particular directories from view as they are not needed and adding new directories with sample code as we go, always maintaining the Troubleshooting and Sandbox directories and the Issues page—the functional “Git Garage” where our students take their first experimental steps with GitHub and reach out for help and create a community.

We find especially compelling the concept of an Issues board that transcends the 15-week semester, and that instead tracks tickets over a course's lifespan, so that if students search on a topic by keyword (like 'for-each-group and XSLT'), they might find needed assistance from past semesters' closed issues. The DHClass-Hub then represents a sustained and managed repository, and not just the ad hoc constructions of separate 15-week semesters. It becomes a continuously accessible “hive mind” that increases in wisdom over time, its content available to anyone who cares to dip beneath the immediately visible surface of the open Issues page. What matters most about our course thus remains fully available and develops as a knowledge bank over time. Rather than being contained by an academic calendar, we can participate in the GitHub community's web of trust. The grade-sensitive and other exchanges that need to be managed in private we handle within the firewall of our university's own Blackboard system, where students privately submit their individual homework assignments and track their course grades. While we recognize that GitHub can manage these things privately, we do not think the best use of GitHub should be primarily for course management, but, rather, for the productive estrangement of learning to think *outside* the course.

Students' collaboration in the social media interface of GitHub, once adopted, can be integrated into their daily digital lives, and the archived documentation of code allows and encourages projects initiated by students in a semester to continue, and even to be passed along to a new group of students the following year, as some of our projects have been extending over time. Such has been the case with our students' Emily Dickinson project, which was launched in the fall 2016 semester and expanded to cover a new series of poems and a complete overhaul of its XSLT coding in spring 2016 (see <http://dickinson16.newtfire.org/>). It has also been the case for Parker's own Restoration of Nell Nelson project (see <http://nelson.newtfire.org/>), which has been continuously transformed in the hands of two distinct student-project teams, and which continues to develop after her April 2016 graduation from the University of Pittsburgh at Greensburg (Lottig et al. [2016](https://link-springer-com.ezproxy.library.ubc.ca/chapter/10.1007/978-3-319-54226-3_15#CR4); Parker et al. [2016](https://link-springer-com.ezproxy.library.ubc.ca/chapter/10.1007/978-3-319-54226-3_15#CR7)).

## Conclusion: Garage Mechanics of the Humanities

In his address about 'The Art of Digital Humanities' at the University of Pittsburgh on 22 March 2016, Stephen Ramsay observed that once the people who worked on pasteurization were not respected as practicing the “real” academic disciplines because what they did looked too much like the work of “mechanics and plumbers” to be taken seriously. His point was that the digital humanities' most significant and lasting contribution might be in giving serious attention to methods that seem the homely handicraft of experimental enthusiasts working outside conventional academic bounds. His “mechanics and plumbers” phrasing struck us at the time as ironically similar to our own affectionate metaphor for our DHClass-Hub as a garage where students and instructors are the mechanics tinkering with one another's code. We practice best, we think, as garage mechanics experimenting with strange new methods of working rather than as traditional professors and students. The major reasons for us to sustain our garage implementation of GitHub indefinitely are for our students to:

* acquire mindful file management skills
* familiarize themselves with the command-line interface
* gain coding and writing skills together in documenting their commits and issues
* recognize the potential for the long-term sustainability of their digital work.

The last goal prioritizes the development of digital repositories, including the community resources of a course, in an openly collaborative environment influenced by semester time-constraints but nevertheless unbounded by them. When considering the goals of our digital teaching practice beyond the temporary interests of a semester class, such repository development can be a key resource to support the long-term development of communities of practicing coders who build sustainable projects.

## Footnotes

1. [1](https://link-springer-com.ezproxy.library.ubc.ca/chapter/10.1007/978-3-319-54226-3_15#Fn1_source).
2. For courses involving applied coding and programming, some instructor-organized discussion and review seems helpful. Skirpan and Yeh held hack-a-thons during class time in their Spring 2014 course, but decided that in future courses they will respond to student suggestions to incorporate an overall discussion and review of common issues and errors before beginning a new hands-on exercise (217).

### References

1. Bergmann, J., & Sams, A. (2014). Flipped learning: Gateway to student engagement. *Learning & Leading with Technology*, *41*(7), 18+.[Google Scholar](https://scholar-google-com.ezproxy.library.ubc.ca/scholar?q=Bergmann%2C%20J.%2C%20%26%20Sams%2C%20A.%20%282014%29.%20Flipped%20learning%3A%20Gateway%20to%20student%20engagement.%20Learning%20%26%20Leading%20with%20Technology%2C%2041%287%29%2C%2018%2B.)
2. Beshero-Bondar, E. (2017). *Digital Humanities Courses*. Retrieved August 21, 2016, from <http://newtfire.org/dh/>
3. Kertész, CZ (2015). Using GitHub in the classroom—A collaborative learning experience. In *2015 IEEE 21st International Symposium for Design and Technology in Electronic Packaging (SIITME)* (pp. 381–386). <http://doi.org.ezproxy.library.ubc.ca/10.1109/SIITME.2015.7342358>
4. Lottig, N., Stewart, B., Mielnicki, A., & Lawrence, B. (2016). *Emily Dickinson*. Retrieved August 21, 2016, from <http://dickinson16.newtfire.org/>
5. Moretti, F. (2013). *Distant reading*. London: Verso Books.[Google Scholar](http://scholar.google.com.ezproxy.library.ubc.ca/scholar_lookup?title=Distant%20reading&author=F.%20Moretti&publication_year=2013)
6. Parker, RJ (2016). *Guide to git shell: The basics of operating GitHub using command line*. Retrieved August 21, 2016, from <http://newtfire.org/dh/explainGitShell.html>
7. Parker, RJ, Spadafore, R., et. al. (2016). *The restoration of Nell Nelson: An investigation of the Chicago Times. Series “City slave girls”*. Retrieved August 21, 2016, from <http://nelson.newtfire.org/>
8. Ramsay, S. (2012). Programming with humanists: Reflections on raising an army of Hacker-Scholars in the digital humanities. In *Digital Humanities Pedagogy: Practices, Principles and Politics*. Retrieved from [http://www.openbookpublishers.com/htmlreader/DHP/chap09.html#ch09](http://www.openbookpublishers.com/htmlreader/DHP/chap09.html%23ch09)
9. Skirpan, M., & Yeh, T. (2015). Beyond the flipped classroom: Learning by doing through challenges and Hack-a-thons. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education* (pp. 212–217). New York, NY, USA: ACM. <http://doi.org.ezproxy.library.ubc.ca/10.1145/2676723.2677224>