Digital Humanities Development without Developers: Bulgarian Dialectology as Living Tradition

Quinn Dombrowski
Research IT
U.C. Berkeley
Berkeley, CA
quinnd@berkeley.edu

Ronelle Alexander
Dept of Slavic Languages & Literatures
U.C. Berkeley
Berkeley, CA
ralex@berkeley.edu

ABSTRACT
In this paper, we use *Bulgarian Dialectology as Living Tradition* as a case study in a low-cost approach to digital humanities project development that can provide a highly customized data model and fairly flexible, intuitive user interface, without any custom programming. The infrastructure for this project consists exclusively of general-purpose, open source code (modules) written by an international community of developers for the Drupal content management system. Using this approach, individuals with domain knowledge can develop the data models that underpin the project and configure the system themselves, exclusively by means of Drupal's administrative graphical user interface. This approach is applicable to projects that focus on the curation and presentation of data (i.e. where the technical needs of the research project have close analogs outside the academy). A major barrier to the adoption of this method of developing projects is the widespread belief that the uniqueness of each scholarly research project extends beyond the subject matter to include the project's technical needs as well. This may be the case for research that relies heavily on algorithm-driven analysis, but data curation and presentation are essential in contexts ranging from businesses, government, non-profits, and cultural institutions. The diverse requirements of Drupal's large user base means that Drupal modules are more likely to have been written with an eye towards accessibility, UI/UX design, and mobile optimization than is code written specifically for a scholarly project, where these factors are often deprioritized due to limited resources. Recognizing the inevitable limitations that reliance on pre-built components places on a project, we discuss a number of challenges we encountered. We then compare the basic Drupal content management system to platforms that are scholar-oriented – both those based on Drupal and those using a different code base. Finally we delineate the circumstances under which this development approach is most valuable, and some considerations for adopting it.

General Terms
Documentation, Design, Human Factors, Standardization.

Categories and Subject Descriptors
D.2.13 [**Reusable Software**]: reusable libraries, reuse models.

DH-CASE '14, September 16, 2014, Fort Collins, CO, USA

http://dx.doi.org/10.1145/2657480.2657481

Keywords
Digital humanities, Drupal, sustainability, reuse, open source, data modeling

1. INTRODUCTION
Digital humanities tools and methodologies have seen increasing uptake over the last decade, as evidenced by the growing number of venues for presenting digital humanities work, the appearance of digital humanities topics at traditional disciplinary conventions (e.g. the Modern Language Association, the American Historical Association), and the increasing number of submissions to the international conference of the Alliance of Digital Humanities Organizations[1]. The scope of scholarly activities encompassed by "digital humanities" is extremely broad; some work, such as the application of critical theory to code, requires little by way of supporting infrastructure. However, a large subset of digital humanities work takes the form of "projects", which typically involve the curation, annotation and/or encoding of disciplinary material to facilitate discovery, analysis and/or visualization using newly-developed or existing tools.

Digital humanities projects are time-consuming endeavors, requiring hundreds of hours dedicated to preparing the material under consideration before the scholar can see meaningful results. Institutional structures exist to facilitate the hiring of student assistants who can help relieve the burden of data cleanup, conversion and entry, at relatively little cost to the faculty member. The same cannot be said for hiring individuals with the requisite skills to develop the technical infrastructure for a project. The high costs for hiring a skilled IT architect and programmer(s) to develop a platform for a digital humanities project makes this an unfeasible option for all but the best-funded projects. While some projects can draw in collaborators from computer science due to their computationally interesting problems or use of algorithms, a great deal of field-advancing work can be achieved within individual disciplines by means of common tools and platforms. Furthermore, projects with modest and non-specialized technical needs are at a significant disadvantage when seeking funding, as multiple major grants require "innovation". Although the grant guidelines state that the innovative aspect of a project need not be technical, the number of applications—including proposals for fundamentally new tools and approaches—makes it difficult to compete as a project that aims to apply well-established techniques to a new field or sub-field.

Without large grants to fund development, projects look to free and low-cost services, such as those provided by central IT or the library. For the sake of scalability and low-cost operation,

general-purpose, centrally-provided services for collaboration, file storage, and web publishing lack the customizability or integration with one another to be particularly well-suited to digital humanities project development. While it may be possible to cobble together a project using such tools, their inflexibility limits the kinds of scholarly questions the project can address.

For digital humanities to have an impact on small fields and sub-fields with modest technical needs, scholars need access to a platform that can support the structured input, annotation and linking of any kind of data, as well as multiple reasonably intuitive forms of presentation. Such a platform should have few, and common, technical requirements, and should be compatible with inexpensive shared web hosting. It should be open source, free, and supported by a large community of developers who actively use it themselves. Finally, it should be largely or entirely configurable via a user interface, so that scholars without grants can leverage their own sweat equity to develop the project, without having to develop programming skills first.

It was with these factors in mind that we adopted Drupal as the platform for *Bulgarian Dialectology as Living Tradition*, and argue in favor of a scholar-driven development approach using open source software for similar projects. Our project team, as commonly understood, included no developers. In another sense, by using Drupal, with its modular extensions and active message boards, our project was supported by thousands of developers through the work they were already doing as part of the Drupal community.

1.1 The state of Bulgarian dialectology

Bulgarian dialects have been well studied, the key resource being the four-volume Bulgarian Dialect Atlas (1964-1981), containing some 1200 maps all told. The material is derived from hundreds of hours of field work in 395 villages over a six-year period, using questionnaires that largely solicited individual words. Each of the maps in the atlas is devoted to a particular question; nearly all are focused on phonetics or the lexicon. Information is conveyed through symbols on maps, one for each investigated village. Sometimes the information is binary, with a symbol of one color denoting presence of an element and a symbol of another color denoting its absence; other times a number of options are depicted, with symbols of different colors or shapes representing the different options. "Commentaries" to these maps give supplemental information by listing some of the responses.

The atlases are data-rich, but the organization presents challenges for the scholar who wishes to utilize them. At the word level, one must comb through many different data sets, each constructed in a different manner; and the absence of data above the word level means one cannot study questions of syntax, phraseology, or discourse – all of which require longer stretches of data – at any systematic level. There has been little work done to make this data available in a usable electronic form. An earlier project by one of the authors (Dombrowski) attempted to automatically generate the same maps provided by the atlas by using XML to encode the relevant linguistic traits for each word listed in the commentary. The result showed that the printed maps obscure a great deal of nuance, but the project was abandoned when it became clear that the lack of a uniform questionnaire meant that the data gathered was too inconsistent to support the more interesting intended analyses (e.g. looking at which words were the unique carriers of a given phonological reflex at a given site.)

1.2 Project context

Bulgarian Dialectology as a Living Tradition aims to rectify these shortcomings in the available data by taking a different approach to the gathered field data. Although most dialectologists record speech samples from elderly consultants, transcripts of connected speech are usually published only as "samples" in the appendices to monographs on individual local dialects. By contrast, this project makes such recordings the center of analysis, including audio clips from actual field tapes. Text samples cover the span of Bulgarian dialects; each illustrates not only salient features of the dialect but is also a well-formed narrative, either about individual life histories, or about traditional customs, practices and beliefs, or about how village life is changing. The bulk of the material was recorded in the mid 1990s, within a few years after the fall of communism in Bulgaria. These recordings are now a unique resource, as the generation of individuals who vividly remembered pre-industrial folk culture has largely passed away.

This project is a collaboration between one of the authors (Alexander) and Vladimir Zhobov, who originally envisioned that the final product would take a printed form. Each text selected for inclusion would be transcribed from the audiotape and translated into English. Each word in the text would then be glossed and annotated with grammatical information, and prose commentaries accompanying the texts would describe interesting linguistic and cultural characteristics. Preparing the texts for publication involved breaking each transcribed text into lines that could comfortably fit the width of a page, and presenting each line as a table in Microsoft Word, where the appropriate annotations could be added. In this workflow, Alexander would annotate each word, each time it appeared. This was not only time-consuming, but it also increased the likelihood of inconsistencies in the data, as a result of annotating the same word in the same context in two different ways (e.g. "nominative singular" vs. simply "singular").

At a national Slavic conference in January 2011, Dombrowski (who holds an MLA in Slavic linguistics, in addition to an MLS) proposed reconceiving the project as a database. Each annotated word would exist a single time in the database, and lines could be constructed via pointers to those annotated words. A database approach would also easily support other ways of representing the data beyond those that had been selected for the print version. For instance, a "reading version" that contained only the English and/or Bulgarian text would be relevant for the general public, or scholars who were only interested in the content of the narratives. Stripping out the linguistic annotations to provide such a version in print would be extremely time-consuming and tedious, but doing it from a database would be trivial. Adding a small bit of additional metadata – geographic coordinates for the villages, and corresponding "dictionary" forms for each dialectal word – would also enable interactive map- and glossary-views of the data that more closely parallel the traditional publication form for Bulgarian dialectal data.

Alexander accepted Dombrowski's offer to develop a prototype database using some sample data that had already been prepared in Microsoft Word. To accomplish this, Dombrowski made use of the Drupal content management system.

1.3 Drupal

Drupal, an open source content management system written in PHP, was developed by Dries Buytaert, a student at the University

of Antwerp, in 2000[1]. Drupal gained attention in 2003 as the platform behind "DeanSpace", an unofficial network of websites supporting the candidacy of Howard Dean for President of the United States[2]. Subsequently, Drupal has seen widespread adoption among media companies, cultural heritage institutions, government, and higher education administration[2].

Drupal is architected to be extremely modular. Because the needs of the sites that Drupal can support vary greatly, the core Drupal code by itself has very limited functionality. Community-developed add-ons ("modules") provide additional features, and the selection, combination, and configuration of modules gives a particular site its shape. As of May 2014, over 26,000 free modules are available, and over 33,000 developers have committed code to Drupal[3].

Since Drupal 4.6 (2005), modules have been available for site developers to define their own content types, specifying how many, and what kind, of fields would be associated with each content type[3]. This was incorporated into Drupal core code as of Drupal 7. Another module, Views[4], essentially provided a UI for creating SQL queries, allowing users to choose what fields to display, indicate how results should be sorted and filtered, and specify joins and arguments. This set Drupal apart from other content management systems at the time, such as WordPress, which were designed to store blocks of textual content or multimedia, and present that content via individual web pages or as a "content stream" blog interface. Storing data in any arbitrarily structured way, and presenting that data appropriately, was much more difficult than doing so in Drupal.

Major version releases of Drupal are not backwards-compatible[4], which allows for significant changes to the architecture of the system between versions. Major web accessibility improvements, for instance, were incorporated into core code as part of the Drupal 7 release. Drupal 8, which has been in development since March 2011 and has no set release date as of May 2014, has been rewritten to use the Symfony framework, to make Drupal development more intuitive for programmers who are unfamiliar with Drupal's idiosyncratic practices, by providing a standard MVC architecture, object-oriented programming, dependency injection, and YAML for configuration management [5].

Drupal 7 was released in January 2011, just as *Bulgarian Dialectology as Living Tradition* pilot development was beginning. At that time, there was a rich ecosystem of well-supported modules for Drupal 6, while essential modules like Views were only available as alpha releases for Drupal 7, or were not available at all. For that reason, we chose to build *Bulgarian Dialectology as Living Tradition* using Drupal 6. This proved to be a reasonable decision; it took a year for a stable version of Views to be released for Drupal 7. Two major release series of Drupal are supported at a given time[5], and there have continued to be updates for Drupal 6 core and contributed modules, and Drupal 8 will include the ability to upgrade directly from Drupal 6[6].

Drupal has a long history of incorporating innovative features originally developed by its large and active developer community. The Drupal.org site not only contains infrastructure for reporting bugs and contributing patches, but it also provides community-powered support forums for users at all levels. Using Drupal and existing modules to develop a digital humanities project can provide a better sustainability story than having a programmer write custom code. Even if some coding is needed in the future to support the project in the future (e.g. to upgrade an important module for compatibility with a new version of Drupal), that code can be reused by other Drupal sites.

2. BULGARIAN DIALECTOLOGY AS LIVING TRADITION

2.1 Architecture and Content Types

To support the presentation of ancillary material (project overview, an "about" page, news etc.), we kept the "page" and "story" content types that are provided by default in Drupal 6. For the core functionality of the *Bulgarian Dialectology as Living Tradition* site, we developed the following set of granular content types that would be connected to one another via a set of pointers. All configuration was done through the Drupal administrative UI, which allows users to create content types and add new fields by filling out a series of web forms. The connections between content types are illustrated in figure 1.

The data model developed for the content types aimed for normalization where it would add utility for the scholars, while keeping in mind the preferences of the users given the context of this particular data set and project. Dombrowski developed the initial data model, and Dombrowski and Alexander jointly refined it after discussing the implications for both data entry and site functionality. We were not intending for the data model itself to be reusable; all the code for the project was already being reused, and to facilitate both data entry and development of various presentation layers, we chose to have a data model specific to this project and data set. This led the seemingly unorthodox decisions indicated below.

2.1.1 Location

This content type stores information about a site where fieldwork was conducted. Its fields are:

- Location name (required; text)
- Geographic coordinates (longitude and latitude)
- Date visited (in a structured format)
- Region (selection from a pre-defined vocabulary; autocompletes)
- Notes (text box).

Each location was visited as part of a single trip, which meant that the date of the visit made sense to the scholars doing the data entry as part of the information about the location itself.

[1] https://drupal.org/about/history

[2] Dries Buytaert maintains a list of sites that use Drupal, complete with screenshots, on his personal blog at http://buytaert.net/tag/drupal-sites.

[3] Flexinode (https://drupal.org/project/flexinode) was the original module that allowed non-programmers to create content types. This was superseded by CCK (https://drupal.org/project/cck), which was incorporated into Drupal core in Drupal 7.

[4] https://drupal.org/project/views

[5] https://drupal.org/documentation/version-info

[6] https://drupal.org/node/2150405

2.1.2 Informant

This content type stores information about each individual who participates in the dialog. Its fields are:

- Location (a pointer to a node previously created using the "location" content type; autocompletes)
- ID (required; typically a letter, or the initials of an investigator)
- Age (integer)
- Gender (M / F)
- Notes (text box)
- "Investigator" checkbox (allows the user to indicate that this person is an investigator; this enables filtering of word lists to exclude those spoken in the standard language by the investigators)

The age of the informant refers to their age as of the date the recording was made. That date is indicated in a field in the Location content type.

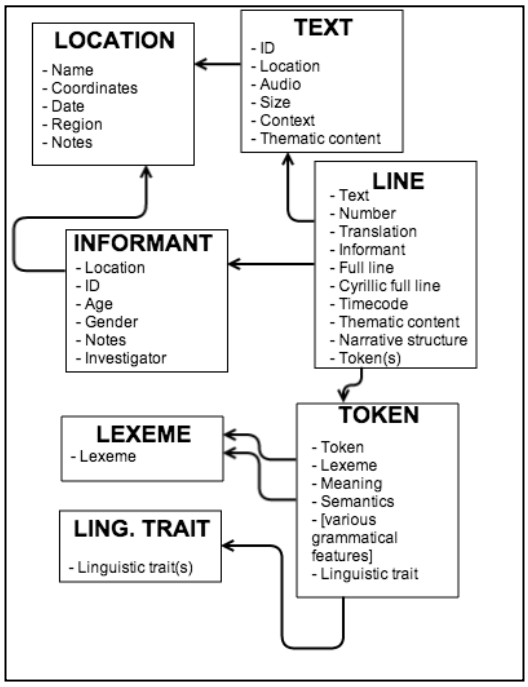

Figure 1. Content types

2.1.3 Text

This content type includes only metadata about the text, rather than the text itself. Its fields are:

- Numerical identifier (required, integer; each text is uniquely identified by the combination of a location and this identifier number)
- Location (required; a pointer to a location node; autocompletes)
- Audio (file upload field for an mp3 of the recording of the text)
- Text size (text field; usually includes the number of lines and the duration of the recording.)
- Physical context of the recording (a text box with a brief description; e.g. "Inside informant's home")
- Brief synopsis of the thematic content (a text box with a brief description).

For administrative purposes, fields indicating the completion status of data entry for the text were added later.

We discussed at some length how to structure the "text size" field. We did consider creating an integer field for the number of lines, and an additional separate field for the timecode, on the thought that this would allow users to sort by text length. Ultimately, we decided that sorting by text length would not be something that our intended audience would be sufficiently interested in doing to merit creating two fields and entering the data in that way.

2.1.4 Lexeme

This content type is used to store dictionary headwords corresponding to each word as spoken in the text. Its fields are:

- Lexeme (required, text; the lexeme itself)

2.1.5 Token

This content type stores words as spoken in the text, and associated linguistic information. Its fields are:

- Token (required, text; the word itself)
- Lexeme (pointer to the dictionary headword corresponding to the word's morphology; autocompletes)
- Semantics (pointer to a dictionary headword corresponding to the word's meaning, if different than the headword indicated by the 'lexeme' field; autocompletes)
- Meaning (text; an English gloss)
- A variety of fields corresponding to grammatical traits (case, number, gender, definiteness, person, verb form, particle, clitic, deictic, pragmatic, other), each with radio buttons for selecting a trait, or N/A

An additional "linguistic trait" field was originally implemented by means of assigning any number of tags from a pre-defined vocabulary (e.g. /dž/ > /ž/, double accent, etymological /ja/, stressed vowel, before palatalizing environment), until it became clear that assigning multiple granular traits as discrete tags with no relationship to one another would make it difficult to perform complex queries. For instance, for a word with an etymological /ja/ and more than one syllable, a tagging system would obscure whether the etymological /ja/ is the stressed vowel, or if the "stressed vowel" tag refers to some other vowel.

To address this issue, a new content type for "linguistic trait" was created. This content type is essentially a compounding mechanism for individual traits (which are still stored in a vocabulary, albeit in their most decomposed form), allowing the user to create traits like "etymological /ja/, stressed, before palatalizing environment"). The token content type has a pointer field to these compound linguistic traits, and a token can be associated with any number of traits.

2.1.6 Line

The line content type is where the textual content is constructed in a recognizable form. Its fields are:

- Text (required; a pointer to the text node the line belongs to; autocompletes)
- Number (required, integer; the number of the line)
- Informant (a pointer to the informant node corresponding to the speaker of the line; autocompletes)
- Token (a pointer to one or more token nodes; autocompletes. Tokens are entered in the order they appear in the line, but can also be rearranged)

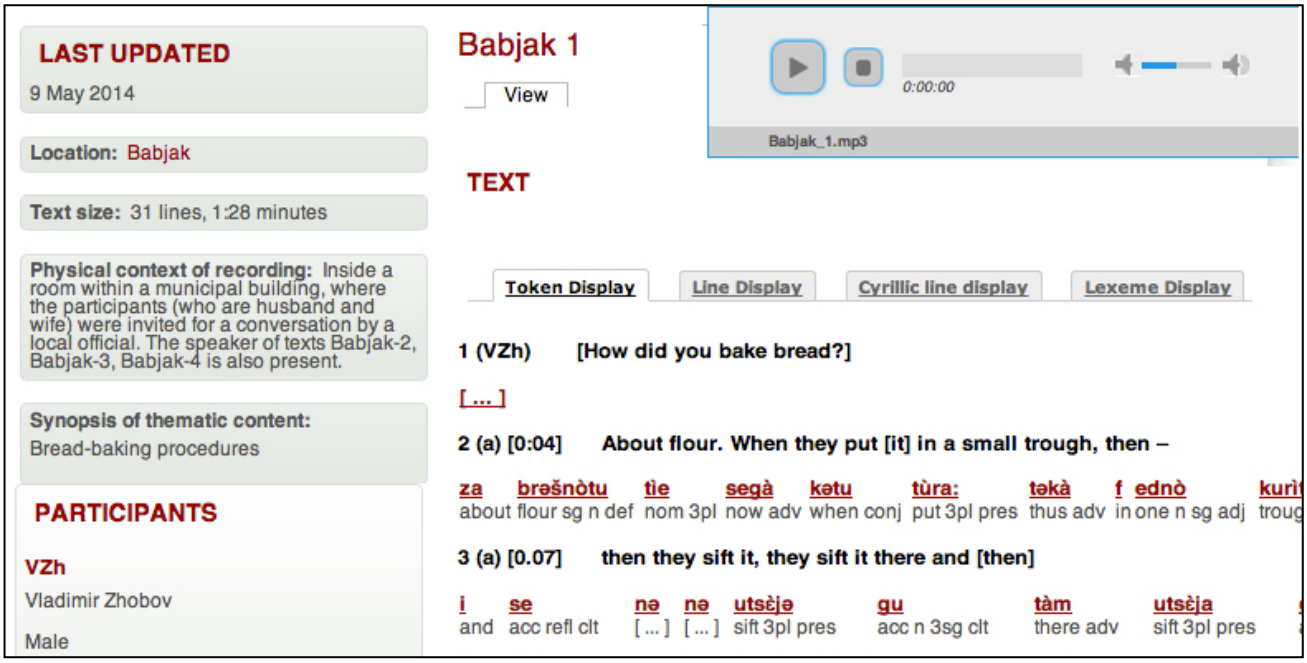

Figure 3. Excerpt from a full text view

- Translation (text field; English translation of the whole line)
- Full line (text field; for presentation purposes, it was easier to have a separate field for the full line, with standard capitalization and punctuation, as opposed to concatenating all the token referents)
- Cyrillic full line (text field; the full line using the Cyrillic alphabet, rather than the transcription into Latin)
- Timecode (text field; the position in the audio recording corresponding to the particular line)
- Thematic content (tagging from a pre-defined vocabulary; autocompletes; examples include 'gender roles', 'collectivization', 'weaving', 'crime')
- Narrative structure (tagging from a pre-defined vocabulary; autocompletes; examples include 'anecdote', 'fairytale', 'proverb')

The inclusion of the two "full line" fields is potentially risky from a consistency standpoint – if any word has to be changed, the change must happen in three different places (the token itself, the full line, and the Cyrillic full line.) In our experience of overseeing data entry, however, there is very little flux on the level of the line; almost all of the changes relate to the metadata associated with the tokens, rather than the tokens themselves.

2.2 Presentation

The *Bulgarian Dialectology as Living Tradition* site has a fairly sophisticated, custom interface that would have required extensive technical resources to develop from scratch. Like the architecture and content types, the interface was built exclusively using Drupal modules that had already been developed. All configuration was done using Drupal's administrative UIs, with the exception of a small amount of CSS customization. No customization was needed on the level of the site's PHP template files.

The interface for the site allows users to navigate the content by 1) selecting a text directly from a list of locations and their texts; 2) browsing a glossary list of tokens, organized by lexeme; 3) browsing for tokens that match certain grammatical traits; and 4) browsing for tokens that match certain linguistic traits, as illustrated by a map provided by a Google Maps module (figure 2).

Figure 2. Linguistic traits view with map

The display of an individual text (figure 3) unifies information stored in multiple content types, drawing from text, informants, lines, and tokens. An audio player floats in the upper right corner, to make it easy for users to start and stop the audio recording, or move ahead to the line they're looking at. Tabs along the top of the interface allow the user to jump to alternative views, including "line display" (the "reading view" without linguistic annotations). Clicking on an individual token takes the user to a page with all the information about that token, including a link to the lexeme, and clickable links to every line where the token appears.

All of the presentation interfaces are generated by the Views module, with supplemental modules providing additional

functionality (e.g. GMap[7] for the Google Map display, Views Display Tabs[8] for the tabbed interface on the text page.)

2.3 Administration and workflows

The location, informant, and text content types are populated using Drupal's default interface for creating nodes (instantiations of a content type). The majority of the content on the site – tokens, lexemes, lines, linguistic traits – is created using a set of interfaces tailored to the data entry workflows of the users, built using the Views and Editview[9], which allows users to create and edit nodes within an interface largely defined by Views. These have been refined as the division of labor among the project team has evolved. Initially, lexeme specification was part of the token editing interface, but once it became clear that specifying lexemes would be a separate task from adding grammatical information, it was moved to a separate editing interface, along with grammatical traits.

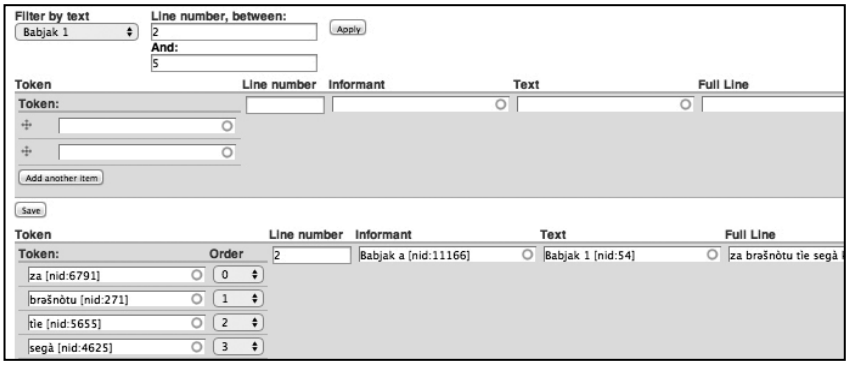

Figure 4. Excerpt from line-entry interface

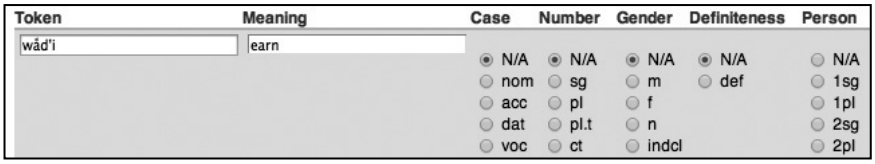

Figure 5. Excerpt from token editing

Currently, text entry is primarily done by undergraduate research assistants who work on the project for credit-hours as part of the UC Berkeley Undergraduate Research Apprenticeship Program. Only one of the student assistants to date has any prior knowledge of Bulgarian, though a few have studied related Slavic languages. Alexander prepares the information for each text in advance, including the linguistic information for each token, and the students enter the data into the system using the line-entry interface (figure 4). If a line requires a token that does not appear in the autocomplete (which should indicate that the token is not yet in the database), the student adds it, and the token automatically appears in an "empty tokens" interface where they can easily add the necessary grammatical information. A similar interface (figure 5) allows users to edit existing tokens.

2.4 Development and Maintenance

Dombrowski completed the core development of the content types (including the prerequisite data modeling work), display views, and administrative views in approximately 15 hours. No custom code was written as part of this process. An additional 2-3 hours were spent customizing the CSS of a theme for the site. Over the course of three years of active site use by Alexander, her Bulgarian colleagues, and student assistants, Dombrowski has spent fewer than 50 hours installing module updates, adding new features, troubleshooting issues, and reworking data models (e.g. the implementation of linguistic traits).

[7] https://drupal.org/project/gmap
[8] https://drupal.org/project/viewsdisplaytabs
[9] https://drupal.org/project/editview

2.5 Challenges and Shortcomings

The initial data modeling for the site was fairly effective in capturing the nuances Alexander wished to use for searching and browsing. This was a relatively lightweight data modeling exercise, well-suited for a team member with a background in library and information science, and familiarity with the subject matter (Dombrowski); see section 3.2 for further discussion of how other projects might undertake this work.

One major shortcoming was in the area of phrases: there is currently no way to provide annotation at the level of a multi-word unit. The planned workaround involves creating a phrase content type that can be associated with an individual line, but this is admittedly awkward. A better integrated approach would require rethinking the line content type, and/or significantly reworking some of the text-display views.

By default, Drupal's autocomplete functionality only shows 10 results. In some cases, previously-entered, correct tokens would not appear in the result list, leading to duplicate entries. A module that allows site administrators to specify the number of autocomplete results has addressed that issue, though cleanup of old redundant tokens is still needed.

Drupal 7 has a number of features not available in Drupal 6 (such as the ability to add fields to taxonomy terms, which would simplify the revised implementation of linguistic traits), but any upgrade is blocked by the Editview module, a crucial component of most of the site's administrative interfaces. The Drupal 6 version is adequately maintained by the developer, but there is no indication that he is working on a Drupal 7 release. Other users of the module have tried for a number of years to hack together a Drupal 7 version, but there is no usable solution. There are a number of Drupal 7 modules that provide broadly similar functionality, but none work as well as Editview to support the administrative interfaces. We are currently considering whether to continue to wait and potentially upgrade directly to Drupal 8 if a suitable alternative becomes available for that version, or to spend the project's very limited funds on hiring a developer to work on a Drupal 7 version of Editview.

2.6 Reception and Impact

We have presented the *Bulgarian Dialectology as Living Tradition* site at a number of Slavic and Balkan conferences. The new form of the project has been exceptionally well received, particularly by colleagues who work with dialectal corpora and other textual material that would benefit from close annotation. Multiple scholars have approached us, asking when this project will be available as a research tool, and how they can develop a similar project.

The project's impact on the scholarly community's understanding of issues in Bulgarian dialectology is already beginning to be felt. Although a great deal of data entry remains to be done, the project's potential, through its combination of new and better-contextualized data, and tools for surfacing correlations that may be analyzed further by individual scholars, has already been demonstrated in scholarly presentations made by both Alexander and Zhobov on basis of the data. While access is currently limited, we plan to make significant areas of the site (http://bulgariandialectology.org) publicly available in fall 2015.

3. SIGNIFICANCE AND REPLICABILITY

Bulgarian Dialectology as Living Tradition demonstrates the viability of a low-cost approach to digital humanities project development that makes such projects viable even in the absence of funding to support custom technical development. As grants become more competitive, approaches that put technical development in the hands of scholars themselves (and/or their non-technical research assistants) may be the most realistic way for digital humanities methodologies to make inroads in small disciplines and sub-disciplines that would benefit from new ways of modeling, disseminating, and analyzing their data, and can leverage the work of open source developer communities to do so.

3.1 Comparison with Scholar-Oriented Platforms

The tendency to prefer systems that have been developed by and for scholars, due to the perception that scholarly data is inherently unique and can't be adequately supported by general-purpose software, is one social impediment to the broader adoption of this approach. Drupal has been leveraged as one of the technical components in scholar-oriented systems, such as OpenScholar[10], an out-of-the-box platform for building "university microsites" without IT support, originally developed at Harvard University. OpenScholar pre-configures content types, views, and administrative settings, but as a result, its applicability is limited to the use cases it was designed for (e.g. individual scholars' websites, course catalogs). Islandora[11] is a platform that combines Drupal with a Fedora repository, originally developed at the University of Prince Edward Island's Robertson Library. Islandora users can download "solution packs" that provide features and configuration tailored to specific use cases (e.g. the Critical Edition and Advanced Critical Edition solution packs are designed for use with TEI-encoded texts) [6].

Even these Drupal-based turnkey solutions have certain advantages over a platform like Omeka. Omeka's functionality is similarly optimized for only a small range of projects (i.e. those focusing on displaying digital collections), and its adoption has largely been limited to higher education, libraries, and cultural institutions. While there are many developers within those communities who work with Omeka, Drupal has benefited from heavy investment by developers in the commercial and government sectors, in addition to those in libraries and higher education. Drupal-derived scholarly platforms can take advantage of that work. Likewise, there have been developers who have built modules for Drupal that specifically address scholars' needs, including a module for managing bibliographic listings[12] and the TEICHI suite of modules for displaying and searching TEI texts[13]. The prevalence of scholar-oriented developers who work with Omeka has led to a number of high-profile plug-ins for that platform (such as Neatline[14], which enables geotemporal storytelling), though some also have Drupal-compatible versions (e.g. Scripto[15], a plug-in for transcription.)

From an architectural angle, a reasonable argument could be made for the approach taken by Islandora, where the peculiarities of an individual text are handled at the level of the TEI encoding, and the platform itself only has to be able to handle TEI—regardless of whether the text consists of ancient Greek inscriptions or modern Spanish electronic literature. Even leaving aside the particular difficulties of processing TEI due to the prevalence of both schema customization and inconsistent application of suggested elements, such an approach is unlikely to work in practice for projects like *Bulgarian Dialectology as Living Tradition*. The expected ways of presenting data in the field of Balkan dialectology are sufficiently different from the presentation forms used in critical editions that one cannot be simply subbed in for the other. Adopting a platform like Islandora with the critical edition solution pack would make the resulting work far less usable by its core audience (scholars of Balkan dialectology), and would require a far more cumbersome TEI-based workflow.

For a project like *Bulgarian Dialectology as Living Tradition*, where the desired presentation is both relatively straightforward and not comfortably matched by a "turnkey" academic solution, taking the Drupal content management system itself as a starting point is better than trying to modify a poorly-fitting turnkey system. The implementation of Drupal-based scholar-oriented platforms varies, but for some—such as OpenScholar—users are essentially dependent on the platform's developers to release a new version that incorporates updates to the constituent modules and Drupal core. When building a site from scratch, even if one module is poorly maintained, the site developer can at least keep all other modules and the core code updated to the latest version, limiting overall exposure to security flaws. The same cannot be said for platforms implemented as a "Drupal distribution", like OpenScholar. The configuration provided by turnkey Drupal platforms tends to be quite intricate, with features well-integrated with one another, and therefore difficult to modify without unintended consequences. Starting with a plain installation of the Drupal CMS provides an opportunity for the site developers to document their configuration settings from scratch. Such documentation would not only improve the sustainability of the scholarly project, but it may also be of use to developers of similar projects, who might need to set up a similar display using the Views module.

There are a number of fairly common situations when developing scholarly projects where Drupal currently falls short. Multi-valued fields are trivially easy to set up, but multi-valued field groups (e.g. given name, surname, role – where these three things are connected, and multiple instances of this whole group can be created within a content type) are very poorly supported. Projects that rely heavily on network visualizations or complex statistics would not be well served by Drupal, though the situation could be

[10] https://drupal.org/project/openscholar
[11] http://islandora.ca/
[12] https://drupal.org/project/biblio
[13] http://www.teichi.org/
[14] http://neatline.org/
[15] http://scripto.org/

improved through the development of modules that provide wrappers for JavaScript libraries that provide this functionality.

3.2 Practicalities of Drupal Development for Scholarly Projects

The Drupal content management system is sufficiently complex that either formal training or extensive hands-on experience with existing sites is a prerequisite to building a site that works well. An incomplete understanding of how Drupal content types work, how this should impact data modeling, and how to use the Views module (including its ways of enacting joins and arguments) can lead to configuration choices that will negatively impact the functionality and sustainability of the site, and are difficult to fix later.

There are many books, screencasts, and courses on Drupal, in addition to the official documentation associated with Drupal core and modules. Unfortunately, many of these fall short of sufficiently explaining Drupal's jargon in layman's terms upfront, making the documentation that follows essentially unreadable. The Drupal instructional material written for a general audience generally uses non-academic examples. While only small amount of imagination is required to translate how those examples might work with scholarly data, there is a reasonable case to be made for instructional material tailored specifically to faculty and students considering Drupal as a platform for digital humanities development, both to provide examples that may be more intuitive to less-technical users, and to properly emphasize how to address issues that are common with scholarly data (e.g. managing approximate dates, or BCE dates) but are less relevant in other sectors. Initiatives like *Drupal for Humanists*[16] and the "Drupal for Digital Projects" course at the Digital Humanities Summer Institute in June 2014 are productive steps towards making scholar-led Drupal development a reality.

Librarians may also have a role to play in supporting this approach to project development. Drupal has seen considerable uptake in libraries[17], and library developers may be able to play a technical consulting role for scholars who undertake a Drupal-based project. Even librarians with no Drupal experience may be able to guide scholars developing Drupal-based projects towards best practices (e.g. using existing vocabularies for classifying data, rather than inventing their own lists of terms, factors to consider as they develop the data model for their content, etc).

4. CONCLUSION

In this paper we have shown how Drupal – as a generic content management system, with its broad selection of community-supported modules – is a viable low-cost platform for supporting digital humanities projects. We have illustrated how it can be configured to store scholarly data sets with unique sets of requirements, using only an administrative graphical user interface that is accessible to non-programmers with moderate training. We have described the advantages of this approach, versus adopting and attempting to modify specifically scholar-oriented platforms that do not comfortably meet the project's requirements. Finally, we have argued that this approach to digital humanities project development has the potential to bring digital humanities methodologies to smaller fields and sub-fields, in particular, where the dedication of individual scholars to developing a project using their own time and that of their research assistants greatly exceeds the likelihood of receiving a grant to pay for a programmer's time.

The upcoming release of Drupal 8 may hold great promise or great peril for Drupal as a platform suitable for scholars developing digital humanities projects. The re-engineering of the entire system architecture to better align with professional development practices has already led to friction with the self-identified "hacker" developer community[7], many of whom also work with other platforms used in digital humanities circles, such as WordPress and Omeka. The new architecture of Drupal 8 makes future Scripto-like arrangements (where a plugin was developed both for WordPress and Drupal) less likely. Unless scholar-oriented developers choose to specifically embrace Drupal 8, it may begin to lag behind in areas of scholar-specific module development, and cease to be as appealing a platform for projects like *Bulgarian Dialectology as Living Tradition*. Nonetheless, we maintain that generic, flexible content management systems have great potential for putting digital humanities project development in the hands of individual scholars with time of their own but little funding for technical support, whether or not Drupal continues to fit that profile.

Arguably, the applicability of flexible, modular platforms that can be assembled by less-technical users need not end at those projects that cannot afford their own developer. While it is important that resources be made available to support significant technical innovation, the impact of of "innovation" grants in digital humanities is often curtailed by the lack of funding available for projects that seek to adopt the technical innovation in question. The difficulty of funding such projects allows questions around the true, practical reusability of "innovation" projects at any level—code, architecture, data models, etc. – to go unaddressed. A shift in funding priorities to emphasize the propagation of technical innovation throughout a wide variety of humanistic fields, while holding new "innovation" grant recipients to higher standards in terms of reusability and sustainability (be it through association with an existing platform, modular architecture, etc.), would be a significant step towards making digital humanities methodologies a viable part of any humanist's research toolkit.

[16] http://drupal.forhumanists.org

[17] The American Library Association hosts a Drupal4Lib Interest Group: http://www.ala.org/lita/about/igs/drupal/lit-igdrupal. The group also maintains an active mailing list at http://listserv.uic.edu/archives/drupal4lib.html.

5. REFERENCES

[1] Melissa Terras. 2011. Stats and the Digital Humanities. (November 28, 2011). Retrieved May 27, 2014 from http://melissaterras.blogspot.com/2011/11/stats-and-digital-humanities.html.

[2] Katie Dean. 2003. Netizens Rally for Dean Team. (July 4, 2003). Retrieved May 27, 2014 from http://archive.wired.com/politics/law/news/2003/07/59497.

[3] Drupal. 2014. Drupal homepage. Retrieved May 27, 2014 from https://drupal.org/.

[4] Dries Buytaert. 2006. Backward compatibility. (May 17, 2006). Retrieved May 27, 2014 from http://buytaert.net/backward-compatibility.

[5] Surendra Mohan. 2014. Symfony in Drupal 8. (March 17, 2014). Retrieved May 27, 2014 from http://www.sitepoint.com/symfony-drupal-8/.

[6] Kirsta Stapelfeldt and Donald Moses. 2013. *Journal of the Text Encoding Initiative*. Issue 5, June 2013, article 4. Retrieved May 27, 2014 from http://jtei.revues.org/790.

[7] Jen Lampton. 2013. Introducing Backdrop CMS, a Drupal Fork. (September 16, 2013). Retrieved May 27, 2014 from http://www.jenlampton.com/blog/introducing-backdrop-cms-drupal-fork.