Digital Studies /
Le champ numérique

Research

How to Cite: El Khatib, Randa, et al. 2019. "Prototyping Across the Disciplines." *Digital Studies/Le champ numérique* 8(1): 10, pp. 1–20. DOI: https://doi.org/10.16995/dscn.282

Published: 03 January 2019

Peer Review:

This is a peer-reviewed article in *Digital Studies/Le champ numérique*, a journal published by the Open Library of Humanities.

Copyright:

Open Access:

Digital Preservation:

Digital Studies /
Le champ numérique

El Khatib, Randa, et al. 2019. "Prototyping Across the Disciplines." *Digital Studies/Le champ numérique* 8(1): 10, pp. 1–20. DOI: https://doi.org/10.16995/dscn.282

RESEARCH

Prototyping Across the Disciplines

Randa El Khatib[1], David Joseph Wrisley[2], Shady Elbassuoni[3], Mohamad Jaber[3] and Julia El Zini[3]

[1] University of Victoria, CA

[2] New York University Abu Dhabi, AE

[3] American University of Beirut, LB

Corresponding author: David Joseph Wrisley (djw12@nyu.edu)

This article pursues the idea that within interdisciplinary teams in which researchers might find themselves participating, there are very different notions of research outcomes, as well as languages in which they are expressed. We explore the notion of the software prototype within the discussion of making and building in digital humanities. The backdrop for our discussion is a collaboration between project team members from computer science and literature that resulted in a tool named TopoText that was built to geocode locations within an unstructured text and to perform some basic Natural Language Processing (NLP) tasks about the context of those locations. In the interest of collaborating more effectively with increasingly larger and more multidisciplinary research communities, we move outward from that specific collaboration to explore one of the ways that such research is characterized in the domain of software engineering—the ISO/IEC 25010:2011 standard. Although not a perfect fit with discourses of value in the humanities, it provides a possible starting point for forging shared vocabularies within the research collaboratory. In particular, we focus on a subset of characteristics outlined by the standard and attempt to translate them into terms generative of further discussion in the digital humanities community.

Keywords: software prototyping; interdisciplinary collaboration; standards; geocoding; spatial humanities; shared research vocabularies

In various global contexts, researchers are coming together to imagine the environments that can host, sustain, and facilitate new forms of academic research and inquiry. Indeed, research infrastructures in the academy have evolved to include new spaces of exchange, data management, and computation. At their most virtual, such spaces for digital humanists include software, middleware, cloud computing,

labs, makerspaces and the like. Indeed, we have entered an era of what has been called "Software Intensive Humanities (SIH)," where digital humanists not only use packaged software bundled in their institutional infrastructures, but they also embark on innovative tool creation as a form of generative, critical practice (Smithies 2017). In this article, we explore the idea of proof of concept software prototyping, stemming from a collaboration between researchers in the humanities and computer science, and we examine the issue of the value of collaboration across the disciplines. We have been attempting to model a process that could very well be popularized in coming years, even embedded within basic computational infrastructures for humanists the way that platforms such as Voyant Tools have democratized text analytics. That process is *creating a map from a text.*

Humanities and Computer Science Collaborations: Towards a Product or Prototype?

Digital humanities are a deeply social endeavor, one in which project results are shaped by the various actors involved, as well as by the mutual value drawn by them from the process. Those projects are not always carried out in the context of a shared lab. Cross-disciplinary collaborations also take place virtually, instead of within the confines of a single room, department, university, or even region. Collaboration, it can be argued, is a form of a third place, drawing on the theoretical interests and practical expertise of different kinds of disciplinary actors, taking place in and between their traditional working spaces, and importantly, growing out of the development of shared vocabularies for collaboration and an appreciation of the stakes of the research for others in our team (Bracken and Oughton 2006). Our experience stems from a collaboration initiated informally between a small group of researchers and students in departments of English and Computer Science, rather than within a formal research collaboratory, and at a moment where digital humanities had limited purchase within the home institution.

Whereas interdisciplinarity is an easily promoted ideal, building structures across the disciplines for successful, and sustainable, collaboration is more challenging to achieve (Bos, Zimmerman, Olson, et al. 2007). Collaboration is known to be difficult

for a number of reasons: focus in some disciplines on individual work over research teams, lack of planning and project management skills, lack of infrastructure to facilitate the collaboration, new forms of accountability or communication required to carry projects to fruition, in addition to basic disciplinary difference (Siemens and Siemens 2012). In this article, we turn to another important challenge of collaboration unmentioned above: the ways we characterize the prototype resulting from interdisciplinary collaboration, or in oversimplified terms, the "finished product." Since the work of such software prototyping is iterative, experimental, and without a clear end in sight, the humanists on our team came to appreciate the process as passing through multiple stages of somewhat finished prototypes. A new version or prototype might improve performance or user experience compared with previous versions, but, in turn, eclipsing another part of its previous functionality. We propose to examine in this article how the computational task of text mapping, that is, modelling and operationalizing a relationship between geographic entities and features of language, can be framed within a mutually beneficial language for a collaborative team. We do this by turning to some documentation from beyond the humanities—some might say far beyond the humanities—that, if reframed and generalized, might provide a starting point for forging common goals and vocabulary for interdisciplinary teams. This relies, however, on unpacking, and refining, the notion of the prototype for the specific case of software development within digital humanities.

Software Prototypes: Materializing Contemplative Knowledge

The word *prototype* appears in Renaissance English from a Latinized Greek word meaning a "first form," or a "primitive pattern." A software prototype, according to *A Dictionary of Computer Science,* can be defined as a "preliminary version of a software system in order to allow certain aspects of that system to be investigated ... additionally (or alternatively) a prototype can be used to investigate particular problem areas or certain implications of alternative design or implementation decisions" (n.p.). Prototyping after the digital turn can also be seen as an assemblage of various modes of intellectual work: "*theoria* (or contemplation), *poiesis* (or making), and *praxis* (or practice/action)" (Saklofske 2016, n.p.). According to Saklofske, contemplation

and action are related through the process of making, which can be seen as a materialization of contemplative knowledge both in, and through, engaged activity. Research on building in digital humanities has framed prototyping as an intertwined process of making and thinking, embodied together in the prototype "product;" examples of functional software prototypes, it has been argued, are a contribution to knowledge in themselves in as much as they suggest innovative methodologies (Galey and Ruecker 2010; Ruecker and Rockwell 2010). We assert that a prototype is best understood in a similar light, as intertwined thinking and making, a process of modeling embodied in step-wise software versions (El Khatib forthcoming). In this sense, in Saklofske's terms, the prototype, which embodies the process and product, serves both as an argument and theory. In our case, what kinds of thinking across the disciplines led to our prototype?

Data creation is central to many of our research projects in digital humanities, and it is common knowledge how it can be very time-consuming and expensive. One common research task at the intersection of textual and spatial analysis consists in extracting geographical information from unstructured text and visualizing such data on map interfaces. It is a rather time-consuming process that has led researchers to want to automate the process. Another system that models the text mapping process is the "Edinburgh Geoparser" (Edinburgh Language Technology Group 2017 https://www.ltg.ed.ac.uk/software/geoparser). Practitioners in the spatial humanities have recourse to a growing body of code and critical literature, in addition to infrastructure in the form of gazetteers—digital lists of places against which entities extracted from texts can be matched. Convinced that the immediate linguistic context of geographic entities found in texts is illustrative of the ways that place is constructed by literature, the authors of this paper set out to operationalize this hypothesis by prototyping software named Topotext to carry out the task.

Creating a software prototype involves different skill sets in code, interface design, implementation, and testing; in short, it is a social process. The various iterations of TopoText, from basic conceptualization to implementation, involved different

groups of students and faculty, and this meant that we confronted many disciplinary assumptions that went unmentioned. Furthermore, software prototypes, particularly of the kind one finds in digital humanities, are acts of open scholarship. They are placed in code sharing and versioning environments for others to use, adapt, and refine. Of course, working within a digital humanities lab or on funded projects, one solution to software or coding needs is to contract developers to carry out discrete tasks. If digital humanities enter the research collaboratory, however, where they are face to face with others in computer science (or other disciplines), different dynamics come into play, in which divergent notions of both process and product emerge. Our experience made us very aware of the fact that within the single academic unit of computer science, we also find different forms of reflection and action that map onto the abovementioned axis of theoria, poeisis, and praxis. In other words, "there are many different computings" (McCarty 2005, 158). McCarty qualifies the domain of software *"a locus of confounding"* precisely because he argues, that "the more theoretical side of computer science meets the world through systems engineered to serve and interact with it" (McCarty 2005, 164). Our specific experience has made us see the urgency of thinking through the ways that research is validated from the perspective of different disciplines, as well as within the same disciplinary groupings of the academic unit. At stake here is the way that the common language we might use to describe software prototyping within research teams, and the ways we can take home our results to our different disciplinary homes.

Tensions of Reproducibility

Instead of relying on a service-based, developer-for-hire model of computing, what we call for here is an active discovery of how our disciplinary values and expectations as humanists converge or diverge with those working in different aspects of computing. Within the context of prototyping, we describe a hybrid mode of interdisciplinary academic collaboration set beyond the confines of a physical space (such as a laboratory) or grant-funded project; this collaboration was carried out in its beginning on the same campus, and then subsequently via virtual communication between humanists and computer scientists working from different

sites. The project was experimental, not only regarding the concepts employed, but also in the structure of the collaboration. To our knowledge, no such project had been attempted before between the two academic departments at our institution. In this light, looking back on several years of working together on the TopoText team, we wonder how this type of collaboration was able to pursue its experimental nature while there was no formal support, and likewise, what incentives kept team members pursuing the research project despite the lack of structure. Although they were not clearly articulated in this initial stage of collaboration, there were reasons that the prototyping process appealed to all members of the team. If we are to generalize from the experience, what might be some ways of constructing the frame of collaboration in mutually beneficial ways? How do we balance experimentation and rigor in software prototyping that can bring us closer to "next generation tools" (Siemens, 2016, n.p.)? How can humanists understand what colleagues in computing think is a valuable result in a research project? Some common guidelines would be useful for aligning future collaboration.

Multidisciplinary collaboration models take into account disciplinary characteristics and differences. Major considerations in disciplinary difference include defining research problems and choice of critical vocabulary, designing methodology, asserting authorship, choosing publications venues, assigning rewards and recognition, as well as inter-researcher communication (Siemens, Liu, and Smith 2014, 54). Two models for collaboration in an academic setting are *faculty-oriented research projects* where lead faculty members make decisions on behalf of the entire team and lead the intellectual direction of the project, and *collaboration* that approaches team members as equals, where all members intellectually contribute to the project. The multidisciplinary humanities-computer science team discussed in this paper fits better into the latter, where the students continue to be as invested in its intellectual direction as faculty members. Additionally, it is further from the service-based approached that is more commonly associated with faculty-oriented research projects; here, team members are invested in creating shared research foci, vocabularies, and methodologies.

As would be expected in a new collaboration, when we began building TopoText, the computer science team came to the project with another set of implicit values. Although we worked quite closely together, it is not fair to say that at the beginning of the collaboration, we were completely aware of the other side's workflows or standards of success. One of the members of the team from computer science pointed to systems and software development quality standards, the 2011 International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC), also known as ISO/IEC 25010:2011, a part of the Systems and Software Quality Requirements and Evaluation (SQuaRE), as standards that his field follows when developing software and that informs the pedagogy of software engineering. At first, such a profession-specific document seemed to alienate us, and yet it contained some interesting wisdom; had we not made a conscious effort to "exploit the benefits of diversity" of our project team, we would perhaps have missed this way that his research community articulates project goals (Siemens, Cunningham, Duff, and Warwick 2010, n.p.). Negotiating between the more open-ended, experimental nature of prototyping and the ISO/IEC 25010:2011, indeed involved balancing novelty and conceptual innovation with functional suitability and accuracy. As a standard, it generally challenges the theoretical boundaries of the prototype, while maintaining a level of robustness of the methodologies of the software product. One of the ideas the humanities members of the team had, as we continued to theorize the emerging prototype, was how easy it is for our collaborator's labor and professional expertise to disappear behind the accuracy of the code. In other words, as the software prototype began to embody the qualities of a purposeful, running tool, it was easy to neglect the design decisions and testing that brought the tool into functional existence.

It is important to note that another contributor from the computer science team asserted that he does not rigidly follow such standards as the ISO since the research projects he directs do not have the specific goal of an end-product software system. The divide, if there can be said actually to be one, in our humanities-computer science collaboration was not so much across departmental lines, but

rather between two competing goals, one of experimental modeling without any particular futurity for the prototype in mind, and another that sought to make strategic choices in step-wise implementation planning for scalability and sustainability. We might formulate this insight as a question: for whom does a prototype have an afterlife? The software engineering part of the team described this method of developing code into software as a "spiral," incorporating feedback between developmental phases that allow for modification and improvement. That is to say, new phases are begun before predecessor phases are complete. It would seem, therefore, that unbeknownst to us, the tension in the dual meaning of the notion of the "prototype," signifying both the singular, abstract "original" form and the mold or pattern from which subsequent copies can be developed, was playing itself out in the daily business of our collaboration. The analogy here could be extended to the tension between a "pure" thought experiment, and an experiment with the notion of reproducibility built in to its execution. Reproducibility is key to non-proprietary, open software development these days, as well as to standards of reliability and transparency in certain circles in the digital humanities community, as the use of environments such as GitHub, R Markdown source documents or Jupyter notebooks would seem to attest (Kluyver, Ragan-Kelly, Perez, et al. 2016). The principle of reproducibility also serves historiographic ends, as a way "of thinking-through the history and possibilities of computer-assisted text analysis" (Rockwell and Sinclair 2016). On our TopoText team, there were multiple perspectives on what needed to be done to bring about the software prototype as a public, shareable object: one that conceived the prototype as a kind of *essai*, and another that was conceptualizing the prototype as a structure in ways that it could be expanded on later. Whereas in digital humanities we might speak of operationalizing a concept, that is, translating a theoretical concept into a finite, computable experiment, others aim to move beyond experimental thinking with a computer to build a tool guided by best practices of software development so that it can be shared, distributed and further elaborated. Again, we are reminded of the "rich plurality of concerns" included within computing (Edwards, Jackson, Chalmers, et al. 2013). Let us push

forward to explore, then, the differences and commonalities between such social experiments in digital humanities software prototyping and the abovementioned ISO/IEC 25010:2011 standard.

Forging Vocabularies of Collaboration: From Standards to Guidelines

We are not suggesting that all digital humanities projects or collaboratories should aim for standardized implementation. Far from it. Instead, it is worth examining to what extent the details of the ISO/IEC 25010:2011 *standard* for software development might be translated into *guidelines* for software prototyping. Could delving into these standards be helpful for defining a mutually comprehensible language for some of the guidelines of the research collaboratory? Are there aspects of them that all sides of a collaboratory can share? Are there other aspects that are simply too commercially oriented to take root in the open source ethos espoused by digital humanities? Are there ways that the ISO/IEC 25010:2011 standard might be used to draft some more general guidelines, or even rethought to be useful to the informal, interdisciplinary encounter such as ours? Could such guidelines be scaled from the informal collaboration to the more formalized research collaboratory? We believe that the documentation contains some material for mutual understanding and deserves closer analysis.

Central to multidisciplinary collaboration is developing understanding between disciplines, which can be forged through an understanding of field-specific language and the contexts in which it is being used. L. J. Bracken and E. A. Oughton (2006) identify three main aspects of language that are involved in attaining such an understanding: dialectics, metaphor, and articulation. Dialectics refers to the difference between the everyday use of a word and its expert use, as well as the different meanings that are assigned to the same words by different disciplines. Metaphor, or 'heuristic metaphor,' refers to expressions that push a conceptual understanding by systematically extending an analogy (Klamer and Leonard 1994). A metaphor assumes that those involved in the conversation share the same context before making a conceptual correlation. The final aspect is articulation, which refers

to the process of deconstructing one's disciplinary knowledge in conjunction with the disciplines of collaborators in an attempt to identify the building blocks and employ them towards developing a common understanding. According to Thierry Ramadier (2004), "articulation is what enables us to seek coherence within paradoxes, and not unity" (432). This idea of seeking coherence within paradoxes rather than attempting to reconcile disciplinary differences is what drives us to engage with the language of ISO/IEC 25010:2011. All three aforementioned aspects of language play a crucial role in developing a disciplinary understanding as articulated through these standards. Dialectics is crucial since some of the words used in the standards are familiar either in everyday life (such as "trust" or "comfort") or a humanities context (such as "effectiveness"), but are actually used in a more specialized context in software development. We employ a metaphor in our explanation of the "modularity" characteristic below by drawing an analogy to a wrapper in order to explain how the characteristic functions. Our approach to the language of ISO/IEC 25010:2011 focuses on developing an understanding between its characteristics and humanities concepts rather attempting to reconcile the two.

The above standards related to quality control in software development—originally published in 2011 and reviewed and confirmed in 2017, remain in vigor at the moment of the writing of this article. Although software developers aspire to them, and they are taught as guidelines in computer science programs, like all other standards, more work needs to be done to assess to what extent they are effective or even observed. At first take, adapting guidelines for product-driven industry into the humanities may seem counterintuitive, or even meet with resistance; however, let us not forget that adopting—and reinterpreting—the languages of the different strands of computing runs deep in digital humanities. Practitioners have been both adopting and adapting standards since some of its earliest days. Take, for example, the Text Encoding Initiative, which initially adopted Standard Generalized Markup Language (SGML) as an expression of its metadata, and was then succeeded by Extensible Markup Language (XML) that is still being used today. By adopting robust guidelines for markup of structural and conceptual features of humanities data, the TEI community laid one of the foundations of digital scholarship today. Whereas the

TEI community has created "guidelines" out of what are XML standards, would it not be possible to do the same with software prototyping out of the ISO/IEC 25010:2011 document?

As we mentioned above, our first foray into collaborative software prototyping, TopoText 1.0, was made in an undergraduate software engineering course offered at the American University of Beirut (Lebanon). It might be fair to say that the initial impulse for the collaboration—automating a process of mapping toponyms found in texts and conducting some basic textual analysis around them—was a case study through which undergraduates were exposed to industry-defined standards for high-quality software. This does not mean that such standards were scrupulously followed in the ensuing code, nor that TopoText went through all the series of tests for professional software development, but rather that they were the ideal to which the computing team referred in building the software core. After being exposed to this process model, the humanists on the project team were stirred to explore how such standards were not only product-centered aims, but also how they enriched the conceptualization of code-based work. This led us to ask the question: are standards a conceptual apparatus sitting at the human interface of digital humanists and developers without ever being acknowledged as such?

In the "waterfall model" of software development followed by our colleague in computing, an initial phase deals with the translation of concepts into processes and the articulation of specifications. This phase is followed by one focused on design, consisting of a modular decomposition of the steps of the core process. In these first two phases, the humanists worked closely with the computer scientists to articulate a common vision of the conceptual model. In the implementation phase that followed, this was less the case. In the ensuing testing and validation phases, the humanists stepped back in to confirm to what extent the desired processes were successfully implemented. These phases reinforce the social element of software development, in which "tests of strength" of the project's functionality and usability are carried out. Indeed, the testing phase attempts to compare the "symbolic level of the literate programmer with the machinic requirement of compilation and execution of the software" (Berry 2011, 68).

One of the basic tensions inherent in our process-oriented collaborative model was the language used to describe the resultant system. The digital humanists on the team called the first version of the validated system a prototype, by which we meant an initial step that exposed some of the shortcomings of the text mapping process, whereas the software engineering approach characterized the system as a product. For some on the research team, the partial operationalization of a concept was at stake, a full of implementation of which may never be possible, whereas from the software development angle it consisted of an entire "life cycle" from the taking of specifications to the delivery of a workable system. We can conclude from this difference in perspective that the cycles of labor in prototyping, or perhaps just in research where software development is especially involved, from planning, implementation to validation, are conceived of very differently across disciplines. In retrospect, working together necessitated an understanding of our mutual notions of such phases of labor in research.

In the ISO/IEC 25010:2011 standards documentation, the reader is struck by the language of engineering, functionalism and quality control, a far cry from what most humanists, even digital humanists, deal with every day. The 2011 document in question provides guidelines of what it calls characteristics and sub-characteristics for quality software development. The relevant sections of the document are contained in section 4, Terms and Definitions. Section 4.1 outlines "quality in use" characteristics and sub-characteristics, that is to say, traits of a piece of software that deal with the user experience. Section 4.2 outlines "product quality" characteristics and sub-characteristics, in other words, elements related to how the objectives set out in the design process phase are met by the software prototype. Both of these domains, the role and experience of the non-expert user, and the optimal performance of the tool, were issues of perpetual conversation and debate in our collaboration.

The principles set out in the ISO/IEC 25010:2011 document are not all applicable to the specific case of software development engaged in by the authors of this paper; for example, the principle of *freedom from risk* touches on forms of risk

management that do not come into play with our text mapping tool. Likewise, the principles of *physical comfort* and *security* do not seem immediately relevant, since TopoText creates no particular physical stress and works with open source gazetteers and plain text source material. The risk of compromising the confidentiality or integrity of its users is very low to nil. The same might not be true of other software prototyping endeavors with geo-locating devices or wearable computing that collect data about users or create other physical stress. These notions notwithstanding, the characteristics and sub-characteristics of sections 4.1 and 4.2, both user-centered and function-centered, contain quite a few pertinent concepts worthy of our both attention and contextualization within current conversations in digital humanities. It is with them that we believe bridges of dialogue could be built. Space does not allow us to cover every single one of the themes evoked by the ISO/IEC 25010:2011. Here, we will limit ourselves to a brief discussion of a few of them that are most relevant to our experience within the framework of designing TopoText.

By linking various functionalities together and automating a process, some of the more rigid standards were satisfactorily met in the software prototype; without them, the prototype would not exhibit (in terms of the ISO/IEC 25010:2011) *functional completeness*, that is, the extent to which the software functions matched the outlined objectives. For example, the first version of TopoText aimed to map locations from texts using the Google Map API, and also to carry out what Bubenhofer has called "geo-collocation" [*Geokollokationen*], making spatial association of features with natural language (Bubenhofer 2014, 45–59). This approach encountered problems with erroneous spatial data. In the case of nineteenth-century novels about London, the errors were most often mismatched with other places in the Anglophone world with locations named after the geographies of London. Although this version met the sufficient standards to carry out its functions, it left little space for *effectiveness/accuracy.* We did not know that we would discover something about the qualities of the data we were using—historical literary texts and a contemporary gazetteer—as well as about the processes were attempting to model. There is a growing literature on "failure" in digital humanities and the possibility

of a "failure that works," leaving open the possibility of "uncovering and correcting your mistakes to be an essential part of the creative process, rather than something reserved for hindsight" (Mlynaryk 2016). It is not immediately clear if the software development world would adopt the "working failure" as part of its standard, but the notion does seem to be found lurking within Section 4.2 of the ISO/IEC 25010:2011 about product quality, in as much as *functional completeness* of a software prototype, may be satisfied, but *functional accuracy* or *appropriateness* may not.

Affinities Between Software Prototyping and Digital Humanities

Building on the first instance of collaboration, as well as on a functioning skeleton of the first prototype, in the second version of TopoText, we sought to integrate human judgment into the geocoding process. We changed the reference gazetteer to GeoNames and implemented a basic interface by which a list of potential matching places was produced, a function similar to the Edinburgh Geoparser's capacity to disambiguate with respect to a gazetteer. TopoText adds the function of allowing the user to rank, in an act of close reading, the best match. We also added what the layperson might call functionalities to TopoText, aspects of which are also defined by the ISO/IEC 25010:2011, a selection of the characteristics that the first iteration failed to achieve. These include *modularity, reusability* and *maintainability; compatibility, interoperability,* and *coexistence; functional correctness*, and, from the "quality in use" section, *trust*.

Modularity insists that the implementation of the prototype should be well documented and should be based on wrappers to ensure the feasibility of future enhancement, such as replacing used technologies or integrating different libraries. Moreover, the model should be separated from the view (i.e., from ways of displaying the model) in order to support different types of interfaces for data consumers. Essentially, this quality has to do with separating the content from the form. Being an open data generator, TopoText generates a comma-separated values (CSV) file of the geographic entities included in each text matched with lat-long coordinates, which can then be exported, allowing *reusability* in other environments. It also generates the maps and word clouds of most frequent words in collocation with

particular places, although in a separate browser window. Taken from this angle, the development process has aimed to keep the prototype separate from its form. Although this principle has not translated to a seamless, non-expert user experience with the tool, the notion of open data generation overlaps with the standard of *modularity*. The software engineering focus on the tool adopted an "agile scrum" approach with respect to the various functions; new functions can be added—that is, specific theoretical interventions can be operationalized—under their *modularity* and their consistency with the overall process framework. More work can be done in the case of TopoText with documenting its own interwoven process of design and theory to ensure replicability. After all, a prototype must exhibit *maintainability*, that is, it should always contain the seed of its own improvement.

Reusability is one of the key motivating factors for version 2.0 of TopoText. The question of *reusability* finds its most immediate expression in the tool's function allowing for data import and export of the tool's geo-coded data in a plain, CSV format; in other words, all data generated by TopoText is reusable in other GIS-based platforms. This sub-characteristic closely relates to *compatibility*, which houses the two subcategories of *interoperability* and *coexistence*. Both versions of TopoText were created through a deep remix of existing tools and libraries that are interoperable; in the *theoria* stages of the second iteration, the outside data source that TopoText draws upon was revisited and replaced in order to situate it within the realm of open data further; we switched from Google Maps Engine and Map API to Leaflet (an open source JavaScript library for interactive maps) and GeoNames (an open gazetteer published with a liberal Creative Commons license). As we mentioned before, sometimes a new version of a prototype might improve specific functions at the risk of outperforming previous functions. In fact, future versions of TopoText need to upgrade the visualization of its textual analysis to match the improved level of the geocoding. In sum, *interoperability* was taken into account from the beginning and in a way that would allow us to shuffle the coexisting tools as the prototyping process continued. Nonetheless, a working prototype exhibiting *modularity* exists. It remains a work in progress with its different parts changing incrementally.

Functional correctness refers to the degree to which the prototype "provides the correct results with the needed degree of precision" (See ISO/IEC 25010:2011, section 4.2.1.2. System and Software Quality Models. n.d. https://www.iso.org/standard/35733.html). As we mentioned above, the move away from automatic geocoding toward a semi-automatic, human-in-the-loop process of disambiguation of data not only allowed for more accurate matching of place name with spatial coordinates, but it enacted one of the more interesting human-centered aspects of the ISO/IEC 25010:2011, namely *trust*. Reincorporating human close reading, that is, human judgement about location, obviously slows down the data creation process, but it also serves as a way to peek into the "black box;" this semi-automated approach is meant to mediate between the advantages of automatic parsing, namely speed and scope, and the painstakingly slow process of manual geocoding.

Conclusion

Much is made of the interdisciplinary encounters in the digital humanities lab, in particular, the inclusion of other academic voices from outside the humanities, and yet much more needs to be theorized about the languages of collaborative work, especially if we imagine reaching far beyond the humanities for potential collaborators. Such collaborative work necessarily means venturing into disciplinary conventions and idioms that appear foreign and even alienating. Navigating such radically different discourses is tantamount to analyzing, and even deconstructing, the "boundary-work" of disciplinary construction (Klein 2006, 265–283). We might call it a form of digital humanities translanguaging, moving beyond established academic language systems, in order to draw upon complex semiotic resources for enacting our transdisciplinary research. The examples of the International Organization for Standardization (ISO) document have provided us with some starting points for a dialogue with other disciplines, in what might just be an opportunity to infuse lessons learned from critical digital humanities into a software development model. Experimental prototypes such as TopoText implement any number of important design decisions that are based upon theoretical positions, for example, about

the value of the human in semi-automated computational processes. Although notebooks have not been built for TopoText, as Rockwell and Sinclair suggest, it is perhaps a valuable next step as they document for others how theoretical positions become instantiated in code and then developed towards software. For a third iteration, we plan to continue thinking through the terminology of the standards explored in this article, and about how to continue prototyping across disciplines in a meaningful way, seeking points of interest or overlap between what might appear to be divergent research goals. One of the foci will be on the *usability* and *operability* of the prototype, characteristics which refer to the attributes that make software easy to use and control, in particular for non-expert users. This effort will focus on existing functionalities but will address interface design aesthetics, that, incidentally, are also covered in the standards.

Software is meant for something more than an end in itself. Software developers work on innovating their methods in order to fine tune practical applications. Instead of viewing the standards as a technical and limiting framework, or as a strictly industry-based, product-driven set of rules alien to the type of work carried out in digital humanities, let us continue to think of ways that the standards might be drawn upon as resources to shape critically informed guidelines that will enable next-generation software. The standards can, and should, be approached critically, conceived of as a core part of the prototyping process that allows for future flexibility, given changing project goals or project team members, rather than serving as an ideal to which all products conform.

Competing Interests

The authors have no competing interests to declare.

References

Berry, David. 2011. *The Philosophy of Software: Code and Mediation in the Digital Age*. Houndmills: Palgrave Macmillan Limited. DOI: https://doi.org/10.1057/9780230306479

Bos, Nathan, Ann Zimmerman, Judith Olson, Jude Yew, Jason Yerkie, Erik Dahl, and **Gary Olson.** 2007. "From Shared Databases to Communities of Practice: A

Taxonomy of Collaboratories." *Journal of Computer-Mediated Communication* 12: 652–672. DOI: https://doi.org/10.1111/j.1083-6101.2007.00343.x

Bracken, L. J., and **E. A. Oughton.** 2006. "'What do you mean?' The Importance of Language in Developing Interdisciplinary Research." *Transactions of the Institute of British Geographers New Series* 31(3): 371–382. DOI: https://doi.org/10.1111/j.1475-5661.2006.00218.x

Bubenhofer, Noah. 2014. "Geokollokationen – Diskurse zu Orten: Visuelle Korpusanalyse." *Mitteilungen des Deutschen Germanistenverbandes* 61(1): 45–59. DOI: https://doi.org/10.14220/mdge.2014.61.1.45

Edinburgh Language Technology Group. 2017. "Edinburgh Geoparser." Accessed July 18, 2018. https://www.ltg.ed.ac.uk/software/geoparser.

Edwards, Paul N., Steven J. Jackson, Melissa K. Chalmers, Geoffrey C. Bowker, Christine L. Borgman, David Ribes, Matt Burton, and **Scout Calvert.** 2013. *Knowledge Infrastructures: Intellectual Frameworks and Research Challenges*. Ann Arbor: Deep Blue. http://hdl.handle.net/2027.42/97552.

El Khatib, Randa. Forthcoming. "Collocating Places and Words with TopoText." In *Social Knowledge Creation in the Humanities 2*, edited by Alyssa Arbuckle, Aaron Mauro, and Daniel Powell. New Technologies in Medieval and Renaissance Studies Series. Toronto: Iter Press.

Galey, Alan, and **Stan Ruecker.** 2010. "How a Prototype Argues." *Literary and Linguistic Computing* 25(4): 405–24. DOI: https://doi.org/10.1093/llc/fqq021

International Organization for Standardization. 2017. "ISO/IEC 25010: 2011." Accessed Nov 20, 2018. https://www.iso.org/standard/35733.html.

Klamer, Arjo, and **Thomas C. Leonard.** 1994. "So What's an Economic Metaphor?" In *Natural Images in Economic Thought: Markets Read in Tooth and Claw*, edited by Philip Mirowski, 20–51. Cambridge: Cambridge University Press. DOI: https://doi.org/10.1017/CBO9780511572128.002

Klein, Julie Thomson. 2006. "The Rhetoric of Interdisciplinarity: Boundary Work in the Construction of New Knowledge." In *The SAGE Handbook of Rhetorical*

Studies, edited by Andrea A. Lunsford, Kirt H. Wilson, and Rosa A. Eberly, 265–283. Thousand Oaks: Sage Publications.

Kluyver, Thomas, Benjamin Ragan-Kelley, Fernando Perez, Brain Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, and **Sylvian Corlay.** 2016. "Jupyter Notebooks-a Publishing Format for Reproducible Computational Workflows." In *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, edited by Fernando Loizides, and Brigit Schmit, 87–90. Amsterdam, IOS Press.

McCarty, Willard. 2005. *Humanities Computing*. Houndmills, Hampshire: Palgrave McMillan. DOI: https://doi.org/10.1057/9780230504219

Mlynaryk, Jenna. 2016. "Working Failures in Traditional and Digital Humanities." *HASTAC* (blog). Accessed Nov 20, 2018. https://www.hastac.org/blogs/jennamly/2016/02/15/working-failures-traditional-and-digital-humanities.

Ramadier, Thierry. 2004. "Transdisciplinarity and its Challenges: The Case of Urban Studies." *Futures* 36(4): 423–39. DOI: https://doi.org/10.1016/j.futures.2003.10.009

Rockwell, Geoffrey, and **Stephan Sinclair.** 2016. "Thinking-through the History of Computer-Assisted Text Analysis." In *Doing Digital Humanities: Practice, Training, Research*, edited by Constance Crompton, Richard J. Lane, and Ray Siemens, 9–21 London: Routledge.

Saklofske, Jon. 2016. "Digital Theoria, Poiesis, and Praxis: Activating Humanities Research and Communication Through Open Social Scholarship Platform Design." *Scholarly and Research Communication* 7(2): 0201252, 15.

Siemens, Lynne, and **Ray Siemens.** 2012. "Notes from the Collaboratory: An Informal Study of an Academic DH Lab in Transition." *Paper presented at Digital Humanities 2012 Conference*. Hamburg, Germany, July 18. Published on Implementing New Knowledge Environments Blog.

Siemens, Lynne, Richard Cunningham, Wendy Duff, and **Claire Warwick.** 2010. "'More Minds are Brought to Bear on a Problem': Methods of Interaction and

Collaboration within Digital Humanities Research Teams." *Digital Studies/Le champ numérique* 2(2). DOI: https://doi.org/10.16995/dscn.80

Siemens, Lynne, Yin Liu, and **Jefferson Smith.** 2014. "Mapping Disciplinary Differences and Equity of Academic Control to Create a Space for Collaboration." *Canadian Journal of Higher Education* 44(2): 49–67.

Siemens, Ray. 2016. "Communities of Practice, the Methodological Commons, and Digital Self-Determination in the Humanities." *Digital Studies/le Champ Numérique*. DOI: https://doi.org/10.16995/dscn.31

Smithies, James. 2017. *The Digital Humanities and the Digital Modern*. London: Palgrave Macmillan. DOI: https://doi.org/10.1057/978-1-137-49944-8

How to cite this article: El Khatib, Randa, David Joseph Wrisley, Shady Elbassuoni, Mohamad Jaber and Julia El Zini. 2019. "Prototyping Across the Disciplines." *Digital Studies/Le champ numérique* 8(1): 10, pp. 1–20. DOI: https://doi.org/10.16995/dscn.282

Submitted: 24 July 2017 **Accepted:** 15 June 2018 **Published:** 03 January 2019