DHQ: Digital Humanities Quarterly
2014
Volume 8 Number 4

Beyond Gutenberg: Transcending the Document Paradigm in Digital Humanities

David Schloen <dschloen_at_uchicago_dot_edu>, University of Chicago
Sandra Schloen <sschloen_at_uchicago_dot_edu>, University of Chicago

Abstract

Computer-aided research in the humanities has been inhibited by the prevailing paradigm of software design in humanities computing, namely, the document paradigm. This article discusses the limitations of the document paradigm and contrasts it with the database paradigm. It describes a database-oriented approach that provides a better way to create digital representations of scholarly knowledge, allowing individual observations and interpretations to be shared more widely, analyzed more effectively, and preserved indefinitely.

Introduction

Computer-aided research in the humanities has been inhibited by the prevailing paradigm of software design in humanities computing (or, as it is now called, digital humanities). The prevailing paradigm is so pervasive and has been entrenched for so long that many people have difficulty imagining an alternative. We are referring to the document paradigm of software design, in which the structure of pre-digital documents is the basis for the data structures in which information is represented digitally. In the document paradigm, the digital representation of information depends on the relative position of units of information in one or two dimensions. Information is represented by linear character strings or by tables consisting of rows and columns, as on a flat printed page. 1

These position-dependent data structures are familiar to people who have been trained to read books, which accounts for their popularity. But they fail to meet the needs of scholars if they are used, not simply as intuitive display formats, but as the basic organizing structures for the digital representation of scholarly information. We pay a steep price for clinging to convenient analogies like “line,” “ledger,” “page,” “book,” and “library” insofar as we treat them as basic structures in our software rather than merely convenient ways to present information to end-users. By organizing information within position-dependent data structures limited to one or two dimensions, such as strings and tables, we fail to represent the full range of scholarly observations and interpretations in a predictable and semantically rich digital form that permits powerful automated comparisons and analyses.[1] 2

For example, in the document paradigm a text will be represented as a long sequence of numeric codes in which each code represents a character in a writing system (e.g., the standard ASCII numbers used for characters in the Latin alphabet). Some of the characters may constitute annotations or “markup tags” interspersed among characters that represent the text itself, but both the text and its markup will be represented by a single string of character codes. A character string with embedded markup tags can be interpreted as a hierarchical “tree” of textual components, yielding a more complex data structure. However, the semantic potential of the tree structure will be limited by the fact that the markup tags occur at particular positions within what is, ultimately, just a one-dimensional sequence of character codes. 3

This widely used method of text encoding is sufficient for many purposes in daily life and in research outside the humanities. But it has been borrowed inappropriately from non-scholarly domains to be used for literary and historical texts that are objects of study in their own right. Some kinds of scholarly work can be done using this method, but it 4

imposes unnecessary limits if it is used as the primary means of representing a text. It deprives scholars of the power to express in digital form many of the conceptual distinctions they employ routinely in the course of their work. They end up imitating the position-dependent structure of a pre-digital medium rather than exploiting the potential of the digital medium to represent their information in a more effective way. As a result, they fail to capture in an explicit, searchable form the different ways a given text has been read and annotated, making it difficult to use computational methods to compare and analyze the various interpretations. And it is precisely those texts which are open to many different readings that are of greatest interest to scholars in the humanities, who spend considerable time tracing the history of textual interpretation and the interconnections within and among texts — tasks that could be greatly facilitated by the appropriate digital tools.

5 The alternative to the document paradigm is the database paradigm, which is characterized by data structures that transcend the position-dependent structure of pre-digital documents. Database systems make use of unique “keys” and internal indexes to retrieve and recombine atomized units of information in a flexible manner. The database paradigm has been wrongly neglected in the humanities because it is thought to be unsuitable for free-form texts. Databases are thought to be suitable only for highly structured tables of data, which is what many people think of when they hear the word “database.” But there is a great deal of predictable structure both within texts themselves and within scholarly analyses of them — structure of a kind that is best represented digitally by means of a properly atomized and keyed database.

6 Moreover, a data table is itself a document and not a database. Even if we were to aggregate separate tables into a collection of tables, we would not thereby create a database, properly speaking, regardless of whether we use database software to manage the various tables. According to the document-versus-database distinction employed in this article, a collection of tables would constitute a database only if the tables were linked via key fields and were “normalized” to minimize the duplication of information, necessitating “joins” among two or more tables to produce a dynamic view of the atomized information in response to a particular query.[2] Thus, the distinction between documents and databases is not a distinction between unstructured textual information and structured tables. Rather, it is a distinction between position-dependent data structures that mimic pre-digital documents — be they one-dimensional character strings or two-dimensional tables — and the atomized, flexible, and hence multi-dimensional structure of a true database.

7 Unfortunately, because of the pervasiveness of the document paradigm, which has become a deeply engrained tradition in humanities computing, scholarly software development has focused on digital documents. There are a few exceptions but on the whole there has been very little effort expended to develop database systems for research in the humanities, even though there has been no insuperable technical barrier to doing so in the past forty years, since the advent of the relational data model and general-purpose database querying languages. Moreover, in the last ten years, non-relational data models and querying languages have emerged that make possible the creation of powerful database systems that preserve the best features of relational systems but can more easily accommodate texts.[3] It is time to embrace the database paradigm in digital humanities and invest more effort in developing software within that paradigm instead of within the document paradigm. This is the best way to create digital representations of scholarly knowledge that can store individual observations and interpretations in a manner that allows them to be shared more widely, analyzed more effectively, and preserved indefinitely.

8 In the remainder of this six-part article we will explain in more detail why the database paradigm is better for digital research in the humanities than the document paradigm. We will start in Part One by discussing the nature of digital texts and the “problem of overlapping hierarchies” that plagues the widely used linear-character-sequence method of digitizing texts. This text-encoding method conforms to the document paradigm but it cannot cope with multiple readings of the same text, which are best represented digitally as overlapping hierarchies consisting of the same textual components. The inability of this method to represent overlapping hierarchies limits its usefulness for scholarly research.

9 In Part Two, we examine how it was that the document paradigm came to dominate software design in the humanities, despite its limitations. We will summarize the parallel histories of the document paradigm and the database paradigm since they emerged in the 1960s in order to understand the difference between them. And we will see how, in recent

years, the document paradigm has enriched the database paradigm and made it even more suitable for computational work in the humanities.

10 In Part Three, looking forward, we discuss in general terms how to transcend the document paradigm and work within the database paradigm in a way that enables scholars to represent all the entities, properties, and relationships of interest to them, including multiple readings of the same text that are represented by means of overlapping hierarchies. This requires an "item-based" atomization of information. A suitably designed database system that implements an item-based ontology[4] can allow us to represent scholarly knowledge in a highly flexible but still predictable form — a digital form of knowledge whose schema is sufficiently rich, semantically, to permit efficient automated searches, while still making it easy for researchers to reconfigure their data and to integrate it tightly both within and across individual research projects.

11 In Part Four, we describe a specific example of how this atomized item-based ontology has been implemented in a working database system in use at the University of Chicago. This is a multi-user, multi-project, and highly scalable non-relational database system that takes advantage of recent innovations in database software. It relies on a standardized data format (XML) and corresponding querying language (XQuery) that have been incorporated into high-performance database software over the past decade as an alternative to relational tables and the SQL querying language. We have therefore been able to borrow many of the best features of older relational database systems while working more easily with the complex overlapping hierarchies that characterize data in the humanities.

12 In Part Five, we explain how a hierarchical, item-based ontology implemented in a database system allows us to go beyond isolated data silos to a sustainable online research environment that encompasses and integrates a vast array of information of interest to scholars — information that can be rigorously searched and analyzed, in the spirit of the Semantic Web, overcoming the limitations of the traditional document-oriented (and semantically impoverished) World Wide Web. Texts and the many ways of reading them are only one example of the information we need to search and analyze. The same generic ontology can be used for other kinds of information pertaining to persons, places, artifacts, and events. This enables interoperability and economies of scale among a wide range of users. To realize the full benefits of digitization in the humanities, our software tools should allow us to capture explicitly, in a reproducible digital form, all the distinctions and relationships we wish to make, not just in textual studies but also in archaeology, history, and many other cultural and social disciplines. This raises the important issue of "data integration" and how it can best be achieved, which is dealt with in Part Six.

13 In Part Six of this article, we conclude by exploring the implications of the digital ontology we are advocating with respect to the question of whether digitization necessarily forces us to standardize our modes of description or whether we can obtain the benefits of structured data and powerful automated queries without conceding semantic authority to the designers and sponsors of the software we use. We argue that standardization of terminologies and classifications is not necessary for data integration and large-scale querying. A suitably designed item-based system allows us to safeguard the ontological heterogeneity that is the hallmark of critical scholarship in the humanities. Overlapping textual hierarchies are just one example of this heterogeneity, which permeates scholarly work and should not be suppressed by standardized document-tagging schemes or rigid database table formats.

1. The problem of overlapping hierarchies

14 In order to understand why the document paradigm leads to inadequate digital representations of scholarly texts, we must first define what we mean by a "digital text." One way of digitizing a text is to make a facsimile image of it — for example, scanning it to produce a bitmapped photograph — for the purpose of displaying it to human readers. However, a great deal of work in digital humanities is based, not on facsimile images, but on the digitizing of texts in a form that allows automated searching and analysis of their contents. Yet, when we go beyond making a visual facsimile of a text and try to represent its meaningful content, then what we are actually digitizing is not the text per se but a particular reading of it.

15 The standard method of creating a digital text from a non-digital original involves the character-by-character encoding of

the text as a sequence of numbers using one number per character. But this kind of encoding represents just one potentially debatable interpretation of the physical marks on an inscribed medium. Even in the simplest character-by-character encoding of a text, without any editorial annotations in the form of embedded markup, choices are being made about how to map the marks of inscription onto standard numeric codes that represent characters in a particular writing system. Moreover, all character-encoding schemes, such as ASCII and Unicode, themselves embody prior interpretive choices about how to represent a given writing system and, crucially, what to leave out of the representation. These choices were not inevitable but must be understood historically with respect to particular individuals and institutional settings. There is a historically contingent tradition of encoding characters electronically that began long ago with Samuel Morse and the telegraph, with the long and short signals of the Morse code.[5] When we speak of digital texts we should remember that every encoding of a non-digital work is a reductive sampling of a more complex phenomenon. The original work is represented by sequences of binary digits according to some humanly produced and potentially debatable encoding scheme.[6]

For example, the Unicode Consortium makes debatable choices about what to include in its now ubiquitous character-encoding standard [http://www.unicode.org]. Even though it gives codes for many thousands of characters, it does not attempt to capture every graphic variant, or allograph, of every character in every human writing system. This is understandable as a practical matter but it creates problems for researchers for whom such variations are themselves an important object of study because they indicate scribal style, networks of education and cultural influence, diachronic trends, and so on. So we should start by acknowledging that digitally representing a text as a sequence of standard character codes really represents just one possible reading of the text; and, furthermore, this representation is encoded in a way that reflects a particular interpretation of the writing system used to create the text in the first place. 16

The problem is that many texts are open to multiple readings, both on the epigraphic level and on the grammatical or discourse level, and yet the dominant text-encoding method in use today, which is characterized by linear character-code sequences with embedded markup, is not capable of representing multiple readings of the same text in a workable manner. This is a serious problem for scholars in the humanities, in particular, for whom the whole point of studying a text is to come up with a new or improved interpretation of it. It is an especially pressing concern in fields of research in which divergent readings occur at quite low levels of grammatical interpretation or epigraphic decipherment. For example, when working with writing systems that do not have word dividers, as is common in ancient texts, we often face ambiguous word segmentations. And many texts have suffered physical damage or are inscribed in an ambiguous way, giving rise not only to debatable word segmentations on the grammatical level but also to debatable epigraphic readings. 17

The problem is perhaps obvious in the case of philological work on manuscripts, in which we need to record the decisions made by editors and commentators in such a way that we can efficiently display and compare different readings of a text whose physical condition, language, or writing system make it difficult to understand. However, the problem arises also with modern printed texts and even with “born digital” texts. Useful work on such texts can of course be done with simple linear character encodings (with or without markup tags) by means of string-matching searches and other software manipulations that conform to the document paradigm. But a one-dimensional encoding cannot capture, in a way that is amenable to automated querying, the decisions made by different scholars as they come up with different ways of analyzing a text and of relating its components to one another or to another text entirely. 18

With the wealth of software tools at our disposal today, scholars should not have to settle for a method of digitizing texts that cannot easily accommodate multiple readings of the same text. They ought to be able to represent in digital form various ways of reading a text while capturing the fact that these are interpretations of the same text. Computers should help scholars debate their different readings by making it easy to compare variations without suppressing the very diversity of readings that motivates their scholarly work. But the only way to do this is by abandoning the deeply engrained tradition of treating digital texts as documents, that is, as data objects which are configured in one or two dimensions. Scholarly texts should instead be represented in a multi-dimensional fashion by means of a suitably designed database system. 19

To understand the limitations of the document-oriented approach, it is worth examining the scholarly markup standard 20

promulgated by the Text Encoding Initiative (TEI), an international consortium founded in 1987 (see http://www.tei-c.org/index.xml; [Cummings 2007]). As we have said, the dominant method of text-encoding for scholarly purposes is not capable of representing multiple readings of the same text in a workable manner. The TEI markup scheme is the best-known example of this method. We cite it here simply as one example among others, noting that most other encoding schemes for textual corpora work in a similar way and have the same limitations. We note also that a number of scholars involved in the TEI consortium are well aware of these limitations and do not claim to have overcome them.

21 A digital text representation that follows the TEI standard contains not just character codes that comprise the text itself but also codes that represent markup tags interspersed within the text. The creators of the TEI scheme and similar text-encoding schemes have assumed that embedding annotations within character sequences in the form of markup tags yields a text-representation that is rich enough for scholarly work. However, as Dino Buzzetti has emphasized, the markup method has significant disadvantages [Buzzetti 2002] [Buzzetti 2009] [Buzzetti and McGann 2006]. As Buzzetti says, drawing on the Danish linguist Louis Hjelmslev's distinction between "expression" and "content," and citing Jerome McGann's book *The Textual Condition* [McGann 1991]:

> From a semiotic point of view the text is intrinsically and primarily an indeterminate system. To put it briefly, there are many ways of expressing the same content just as there are many ways of assigning content to the same expression. Synonymy and polysemy are two well-known and mutually related linguistic phenomena. [Buzzetti 2009, 50]

Thus, in an essay that Buzzetti co-authored with McGann, the following question is raised:

> Since text is dynamic and mobile and textual structures are essentially indeterminate, how can markup properly deal with the phenomena of structural instability? Neither the expression nor the content of a text are given once and for all. Text is not self-identical. The structure of its content very much depends on some act of interpretation by an interpreter, nor is its expression absolutely stable. Textual variants are not simply the result of faulty textual transmission. Text is unsteady, and both its content and expression keep constantly quivering. [Buzzetti and McGann 2006, 64]

The same point could no doubt be expressed in a less structuralist way, but we are not concerned here with the particular literary theory that underlies this way of stating the problem. Regardless of how they express it, Buzzetti and McGann are surely right to say that the difficulties encountered in representing multiple readings of the same text constitute a major deficiency in the markup-based technique commonly used to digitize scholarly texts (see also [McGann 2004]).

22 Moreover, as they acknowledge, this deficiency was recognized very early in the history of the Text Encoding Initiative and has been discussed repeatedly over the years (see, e.g., [Barnard et al. 1988]; [Renear et al. 1993]). Almost twenty years ago, Claus Huitfeldt published a perceptive and philosophically well-informed critique of the TEI encoding method, entitled "Multi-Dimensional Texts in a One-Dimensional Medium" [Huitfeldt 1995]. He described the conventional method of text-encoding in which "a computer represents a text as a long string of characters, which in turn will be represented by a series of numbers, which in turn will be represented by a series of binary digits, which in turn will be represented by variations in the physical properties of the data carrier" [Huitfeldt 1995, 236] as an attempt to represent multi-dimensional texts in a one-dimensional medium.

23 However, despite his accurate diagnosis of the problems inherent in reducing many dimensions to just one, for some reason Huitfeldt did not challenge the basic assumption that "a long string of characters" is the only data structure available for the digital representation of texts. As he no doubt knew, computer programmers are not limited to using one-dimensional structures. The fact that all digital data is ultimately represented by one-dimensional sequences of binary digits is irrelevant in this context. We are dealing here with the logical data structures used by application programmers and by high-level programming languages, not the underlying structures dealt with by operating systems and compilers (a compiler is a computer program that translates source code written in a high-level programming language into lower level "machine code"). As a computer scientist would put it, the power of digital computers stems from the fact that a "Turing machine," defined mathematically in terms of primitive computations on a one-dimensional

sequence of binary digits, is capable of emulating more complex data structures and algorithms that can represent and manipulate information in many different dimensions.

24 Thus, in spite of the deeply entrenched habit of encoding texts as long character strings, scholars should pay more attention to the fact that a digital computer is not necessarily a one-dimensional medium when it comes to the representation of texts — indeed, they will not obtain the full benefits of digitization until they take advantage of this fact. The failure to adopt a multi-dimensional data model explains why, after the passage of more than twenty-five years, no widely accepted way of representing multiple readings of the same text has emerged in the scholarly text-encoding community. By and large, even the sharpest critics of the TEI markup scheme and its limitations have themselves remained within the document paradigm. They have not abandoned the one-dimensional character-string method of representing scholarly texts in favor of the multi-dimensional structures available in a database system.

25 To understand why no workable way of representing multiple readings has emerged, we must examine more closely the TEI Consortium's text-encoding method. The TEI markup scheme encodes an "ordered hierarchy of content objects" (OHCO), where a "content object" is a textual component defined with respect to some mode of analysis at the level either of textual expression or of textual content (see [Renear 2004, 224–225]). Quite rightly, the OHCO model exploits the power of hierarchies to represent efficiently the relationships of parts to a larger whole. For example, a text might be broken down into pages, paragraphs, lines, and words, in a descending hierarchy, or it might be broken down into component parts in some other way, depending on the mode of analysis being employed. Regardless of how they are defined, a text's components can be separated from one another within a long sequence of characters and related to one another in a hierarchical fashion by means of markup tags.

26 The problem with this method is that any one digital text is limited to a single hierarchy, that is, one primary configuration of the text's components. Using standardized markup techniques, a linear sequence of characters can easily represent a single hierarchy but it cannot easily represent multiple overlapping hierarchies that reflect different ways of reading the same text. The TEI Consortium itself provides some simple examples to illustrate this problem in its *Guidelines for Electronic Text Encoding and Interchange*, including the following excerpt from the poem "Scorn not the sonnet" by William Wordsworth [TEI P5 2013, Chapter 20][7]:

> Scorn not the sonnet; critic, you have frowned,
> Mindless of its just honours; with this key
> Shakespeare unlocked his heart; the melody
> Of this small lute gave ease to Petrarch's wound.

This poem could be represented by a hierarchy based on its metrical features, with components that represent the poem's lines, stanzas, and so on. But the same poem could equally well be represented by a hierarchy based on its grammatical features, with components that represent words, phrases, clauses, and sentences.

27 If we survey literary, linguistic, and philological scholarship more broadly, we find many different ways of constructing hierarchical representations of texts, each of which might be useful for computer-aided research on textual corpora. Our different ways of analyzing texts produce different logical hierarchies and we cannot claim that any one hierarchy is dominant, especially if we are trying to create a digital representation of the text that can be used in many different ways.[8] The choice of which analytic hierarchy to use and how the hierarchy should be constructed will depend on the kinds of questions we are asking about a text. For this reason, a fully adequate digital representation would allow us to capture many different ways of reading a text without losing sight of the fact that they are all readings of the same text. Moreover, a fully adequate digital representation would make full use of the expressive power of hierarchies wherever appropriate, but it would not be limited to hierarchies and it would also allow non-hierarchical configurations of the same textual components, without duplicating any textual content.

28 A brute-force solution to the "problem of overlapping hierarchies," as it has been called, is to maintain multiple copies of identical textual content, tagging each copy in a different way. Here is a TEI encoding of the metrical view of the Wordsworth excerpt using the `<l>` tag (one of the standard TEI tags) for each metrical line and using the `<lg>` tag to

indicate a “line group”:

```
<lg>
<l>Scorn not the sonnet; critic, you have frowned,</l>
<l>Mindless of its just honours; with this key</l>
<l>Shakespeare unlocked his heart; the melody</l>
<l>Of this small lute gave ease to Petrarch's wound.</l>
</lg>
```

Example 1.

The grammatical view of the same text would be encoded by replacing the metrical tags with tags that indicate its sentence structure, using the <p> tag to indicate a paragraph and the <seg> tag for grammatical “segments”:

```
<p>
<seg>Scorn not the sonnet;</seg>
<seg>critic, you have frowned, Mindless of its just honours;</seg>
<seg>with this key Shakespeare unlocked his heart;</seg>
<seg>the melody Of this small lute gave ease to Petrarch's wound.</seg>
</p>
```

Example 2.

However, as the authors of the TEI *Guidelines* point out, maintaining multiple copies of the same textual content is an invitation to inconsistency and error. What is worse, there is no way of indicating that the various copies are related to one other, so it is impossible to combine in a single framework the different views of a text that are contained in its various copies — for example, if one wanted to use an automated algorithm to examine the interplay between a poem’s metrical and grammatical structures in comparison to many other poems. 29

Three other solutions to the problem of overlapping hierarchies are suggested in the TEI *Guidelines* but they are all quite difficult to implement by means of textual markup using the standards-based software tools currently available for processing linear character strings. In addition to “redundant encoding of information in multiple forms,” as in the example given above, the following solutions are discussed: 30

1. “Boundary marking with empty elements,” which “involves marking the start and end points of the non-nesting material. . . . The disadvantage of this method is that no single XML element represents the non-nesting material and, as a result, processing with XML technologies is significantly more difficult” [TEI P5 2013, 631–634]. A further disadvantage of this method — indeed, a crippling limitation — is that it cannot cope with analytical hierarchies in which textual components are ordered differently within different hierarchies, as often happens in linguistic analyses.
2. “Fragmentation and reconstitution of virtual elements,” which “involves breaking what might be considered a single logical (but non-nesting) element into multiple smaller structural elements that fit within the dominant hierarchy but can be reconstituted virtually.” However, this creates problems that “can make automatic analysis of the fragmented features difficult” [TEI P5 2013, 634–638].
3. “Stand-off markup,” which “separates the text and the elements used to describe it . . . It establishes a new hierarchy by building a new tree whose nodes are XML elements that do not contain textual content but

rather links to another layer: a node in another XML document or a span of text" [TEI P5 2013, 638–639].

31 The last option, stand-off markup, is the most elegant solution but it is not easy to implement using generally available software based on current markup standards — and it is not, in fact, included in the TEI's official encoding method but requires an extension of it. Many people who are troubled by the problem of overlapping hierarchies favor some form of stand-off markup instead of ordinary "embedded" or "in-line" markup. However, stand-off markup deviates so much from the original markup metaphor that it no longer belongs within the document paradigm at all and is best implemented within the database paradigm. Stand-off markup involves the digital representation of multiple readings of a text by means of separate data objects, one for each reading, with a system of pointers that explicitly connect the various readings to the text's components. But this amounts to a database solution to the problem. The best way to implement this solution is to abandon the use of a single long character sequence to represent a scholarly text — the document approach — in order to take advantage of the atomized data models and querying languages characteristic of database systems. Otherwise, the complex linkages among separate data objects must be laboriously managed without the benefit of the software tools that are best suited to the task. The database approach we describe below is functionally equivalent to stand-off markup but is implemented by means of an atomized and keyed database.[9]

32 Other solutions to the problem of overlapping hierarchies continue to be proposed, some of them quite sophisticated.[10] However, the complex relationships among distinct data objects that underlie these proposed solutions can best be implemented in a database system that does not represent a text by means of a single character string. Such strings should be secondary and ephemeral rather than primary and permanent. Nothing is lost by adopting a database approach because a properly designed database system will allow the automated conversion of its internal text representation to and from linear character sequences, which can be dynamically generated from the database as needed.

33 With respect to the limitations of the TEI encoding method, some have argued that the fault lies with the Extensible Markup Language (XML) standard that prescribes how markup tags may be formatted [http://www.w3.org/standards/xml]. The TEI method conforms to the XML standard so that XML-based software tools, of which there are many, can be used with TEI-encoded texts. But there is a good reason why standardized markup grammars like XML allow only one strict hierarchy of nested textual components. Accommodating overlapping pairs of markup tags within a single character string would create a more complicated standard that demands more complicated software, losing the main benefit provided by the markup technique, which is that it is intuitive and easy to use, even by people who have modest programming skills. If we are going to work with more complex data structures and software, it would be better to abandon the single-character-string representation of a text altogether in favor of a more atomized, and hence more flexible, database representation.[11]

34 Thus, in our view, the TEI markup scheme and other similar text-encoding schemes are useful as secondary formats for communicating a particular reading of a text that has been generated from a richer underlying representation, but they are not suitable for the primary digital representation of a text. For this one needs a suitably designed database system, keeping in mind that with the right kind of internal database representation it is a simple matter to import or export a "flattened" representation of a given text — or, more precisely, of one reading of the text — as a single character string in a form that can be used by markup-based text-processing software. An existing text-reading stored in digital form as a single sequence of characters (with or without markup tags) can be imported by automatically parsing it in order to populate the database's internal structures. In the other direction, a character string that represents a particular reading can easily be generated when needed from internal database structures, incorporating whatever kind of markup may be desired in a particular context (e.g., TEI tags or HTML formatting tags).

35 In summary, we argue that document-markup solutions to the problem of overlapping textual hierarchies are awkward workarounds that are at odds with the basic markup model of a single hierarchy of textual components. Solving the problem will require abandoning a digital representation of texts in the form of long linear character sequences. That approach may suffice for many kinds of digital texts, especially in non-academic domains, but it is not the best method for the primary representation of texts that are objects of scholarly study and will be read in different ways. Digital documents do not transcend the structure of the flat printed page but rely on relative position in one or two dimensions

to distinguish entities and to represent their relationships to one another, instead of making those distinctions and relationships explicit in multi-dimensional structures of the kind found in a well-designed database. What is needed is a database that can explicitly distinguish each entity and relationship of interest, enabling us to capture all the conceptual distinctions and particular observations that scholars may wish to record and discuss.

2. The history of the document paradigm versus the database paradigm

In light of the limitations imposed by the document-markup method of scholarly text encoding, we may ask why this method took root and became so widely accepted instead of the database alternative. To answer this question, it is worth examining the history of the document paradigm of software design in contrast to the parallel history of the database paradigm. The diagram below summarizes the key developments in the emergence of these two paradigms, leading up to the software environment in which we work today. 36

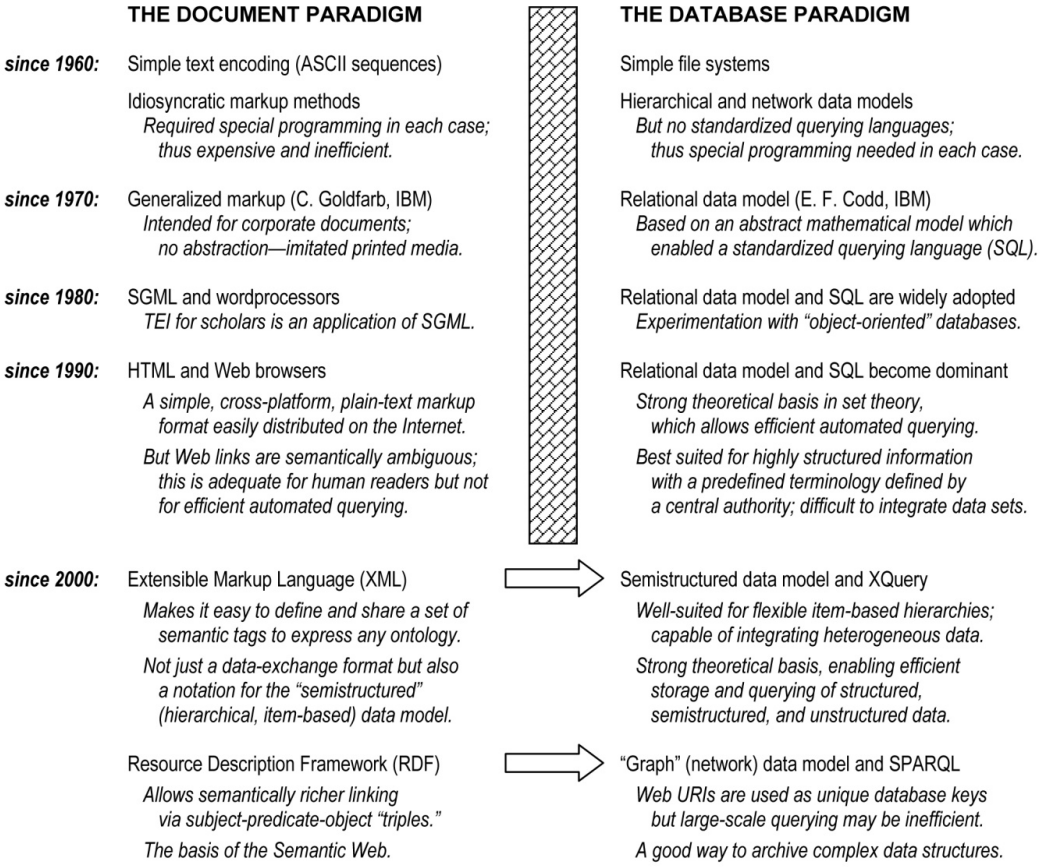

Figure 1.

It is remarkable that current approaches to software design for both documents and databases emerged in the United States in the same year, 1969, and in the same institution, namely the International Business Machines Corporation (IBM), although on opposite sides of the country. At the IBM lab in Cambridge, Massachusetts, Charles Goldfarb and his colleagues came up with the first widely adopted markup method for digital texts. This was eventually canonized as SGML, the Standard Generalized Markup Language, which became an ISO standard in 1986 and a few years later gave birth to HTML, the HyperText Markup Language, and thus to the World Wide Web (see [Goldfarb 1990] on SGML; the relationship between SGML and HTML, and subsequently XML, is described in [DuCharme 1999, 3–24]). 37

Meanwhile, at the IBM lab in San Jose, California, an Englishman named Ted Codd was writing a paper entitled "A Relational Model of Data for Large Shared Data Banks" [Codd 1970], which changed the world of databases forever. Codd's relational data model subsequently became the basis for the majority of database systems in use today. Beyond the relational model itself, the approach to database design that flowed from Codd's conceptual innovations informs 38

many aspects of recent non-relational systems and querying languages.

39 The intellectual contrast between these two IBM employees could not have been greater. Codd was a mathematician who received a Ph.D. from the University of Michigan in 1965 with a thesis that focused on a difficult problem in the theory of computation (self-replication in cellular automata). Goldfarb was an attorney who graduated from Harvard Law School in 1964 and then practiced law in Boston. He did not join IBM until November 1967, at which point, as he says in a memoir, he knew nothing about computers (see [Goldfarb 1996]). He was given the task of applying computers to the needs of law firms, in order to help IBM expand its presence in that market. In particular, he was asked to work with IBM programmers to integrate a simple text-editing program with an information-retrieval system and a text-formatting program to make it easier to manage and distribute legal documents and other structured documents used in business enterprises and governmental settings, such as catalogues and procedural manuals.

40 Given his background, it is not surprising that Goldfarb's solution to the problem was non-theoretical and ad hoc. Quite understandably, he decided to imitate what he knew about the preparation of printed documents and he latched onto the concept of markup, as many others were doing around the same time, transferring the blue pencil-marks of pre-digital editors to special sequences of digitally encoded characters embedded in the main text. These embedded markup "tags" could provide formatting instructions that told a computer program how to print or display a text, or they could provide semantic information about the structure and contents of the text. Goldfarb's innovation was to come up with a standard method for defining new markup tags in a generalized way, allowing anyone to create a set of tags suitable for a particular kind of document and making it easy to write software that could work with a wide range of tagging schemes. His Standard Generalized Markup Language is, strictly speaking, not itself a markup language (i.e., a document tagging scheme) but rather a formal grammar for the design and specification of markup languages.

41 In addition to this standard tagging mechanism, the writing of general-purpose software for text processing was greatly aided by adopting a simple hierarchical model in which tags were used to group textual components into larger configurations in a nested fashion, so that lines of text were grouped within paragraphs, and paragraphs within sections, and sections within chapters, and so on. This simple and intuitive approach to digital documents has proved to be very popular, precisely because of its simplicity. For example, the HyperText Markup Language on which the World Wide Web is based is one such tagging scheme, and it took off like wildfire in the early 1990s in part because it required very little technical expertise to use it.

42 But Goldfarb made a fateful limiting assumption about the nature of texts that has affected digital text-processing ever since. He assumed that the structure of any text that was being digitally encoded would be determined in advance by assigning it to a particular predefined genre ("document type"). This would determine which markup tags could be used and the way in which separately tagged components of the text could be nested within one another to form a single hierarchy. In other words, Goldfarb assumed that someone would decide in advance how the text ought to be read and this way of reading would be fundamental to the text's digital representation. This limiting assumption is not usually a problem in the corporate world in which digital markup emerged, where textual structures are determined from the top down by a central semantic authority. However, this assumption creates severe problems for scholars who lack a central semantic authority and thus work with texts in a different way.[12]

43 Turning now to the database paradigm, we can contrast the ad hoc and non-theoretical character of Goldfarb's approach with Codd's theoretically sophisticated approach to the digital representation of information. Codd's abstract relational model was initially resisted by IBM itself, his own employer, which had recently adopted a quite different model.[13] But the relational model eventually won out and displaced all its rivals. It did so because it is firmly rooted in mathematical theory, enabling the creation of general-purpose software that can search and manipulate relationally organized information with great efficiency. As is often the case in computing, abstract theoretical elegance yielded optimal real-world performance in the long run.

44 Codd's solution to the problem of managing digital information involved a leap of abstraction to the mathematical concepts of "set" and "relation" and the algebraic operations that can be performed on relations (for a concise introduction to the relational data model and relational algebra, see [Garcia-Molina et al. 2009, 17–65]). Codd gave a

mathematical basis to the idea of database “keys,” that is, pieces of information that uniquely identify other pieces of information and can be used to mix and match the information in a database in any number of different ways. By finding the right level of abstraction, he also guaranteed universality, meaning that digital information of all kinds, regardless of which conceptual ontology is used to organize it, can be manipulated and retrieved in a relational database system.

45 Goldfarb failed to make a similar leap of abstraction with the consequence that his method of text representation is not universal but only partial and does not meet the needs of textual scholars in particular. His digital texts mimic the structure of pre-digital documents, conveying information implicitly by means of sequential position, as in a printed work, which conveys information by the relative placement of characters in a line, of lines in a page, and of pages in a book. The information is implicit in the positioning of discrete entities — in their before-and-after juxtapositions with other entities — rather than being made explicit in independently addressable entities that can be related to one another in any number of different ways, as in a keyed and indexed database.

46 In other words, if we rely on data structures that conform to the document paradigm, such as linear sequences of character codes, then our information is imprisoned in the straitjacket of sequential position rather than being free to be configured flexibly in many different dimensions, as can be done in a modern database. This would not matter if all we were trying to do is display a predetermined view of the text to a human reader. But the whole point of the kind of digitization we are talking about is to encode information in a way that enables not just a visual display of the text but automated analyses of its contents. And, as we have seen, what we are actually encoding in that case is not the text per se but one or more interpretations of it. So, if what we want to represent digitally are the ways that one or more readers have understood the text from different perspectives, we must transcend sequential position, which limits our ability to configure a text’s components, and adopt a more flexible, multi-dimensional method of representing a text.

47 It is true that the markup tags embedded in a Goldfarbian digital text allow us to go beyond a representation of the text as a simple unmarked sequence of characters by making explicit some aspects of a reader’s interpretation of the text. But markup tags are limited in what they can do by the very fact that they are embedded in a specific location within a sequence of characters just as pre-digital markup written in blue pencil exists at a certain place on the page.[14] This limitation is tolerable when the structure of the text and thus the possible ways of analyzing it are ordained in advance, as in the world of business and government. But that is not the case in the world of scholarship.

48 The lesson here is that humanists should not be parasitic on the impoverished ontology of digital texts that flourishes in an environment of top-down semantic authority but should draw upon the intellectual tradition of critical scholarship to devise their own richer and more effective ontology. It is unfortunate, in our view, that scholars became reconciled to the limitations of the document-markup approach and neglected the database alternative. However, to be fair, we concede that the relational database software that dominated in the 1980s and 1990s presented some practical barriers for textual applications (e.g., the need for many inefficient table joins). It is only in recent years that these barriers have been eliminated with the emergence of non-relational database systems and querying languages, which themselves have borrowed substantially from SGML and document-processing techniques while remaining within the database paradigm. Thanks to these enhancements, we are at the point where we can stop trying to solve what is actually a database problem by means of a document-oriented method that lacks the powerful techniques implemented in database systems. We can solve the problem of overlapping hierarchies by breaking free of the constraints of the document paradigm and moving firmly to the database paradigm. And we can do so while bringing with us the best features of the document paradigm, which in recent years has greatly enriched the database paradigm and has made it more capable of dealing with relatively unstructured texts.

3. Transcending the document paradigm via item-based atomization of information

49 We turn now from the document paradigm and its history to a database approach that can transcend the limitations of digital documents. But first we must emphasize that the problem we are trying to solve goes beyond the representation of texts, so the best solution will be one that is applicable to other kinds of scholarly data and permits both relatively unstructured texts and more highly structured information to be stored, managed, and queried in the same way. By

generalizing the problem, we can reach the right level of abstraction and develop a common computational framework for many kinds of scholarly research. This allows the same database structure and software to be used for different purposes, yielding financial economies of scale. And the motivation to create a common computational framework is not simply pragmatic. It is also intellectual, because the problem of overlapping hierarchies is not confined to textual studies but emerges in many kinds of research in which multiple configurations and interpretations of the same data must be represented and managed efficiently.

50 The problem we are trying to solve goes beyond text-encoding because the pervasive document paradigm has fostered analogous limitations in humanities software applications developed to manage structured data. Structured data is usually displayed, not as lines of text, but in tabular form, with one row for each entity of interest and one column for each property of those entities. Tables provide a convenient way to display structured data but are not the best way to organize scholarly information in its primary digital representation. A rigid tabular structure imposes predetermined limits on what can be recorded and how the information can be analyzed. In the humanities, there is often a high degree of variability in what needs to be described and a lack of agreement about the terms to be used and how entities should be classified. Software developers who try to cope with this variability by using tables as primary structures will end up creating many idiosyncratic tables which require equally idiosyncratic (and unsustainable) software to be written for each research project, and with no way to integrate information derived from different projects. The alternative we advocate is to exploit the ability of an atomized keyed-and-indexed database system to represent the full range of scholarly observations and conceptual distinctions in a more flexible manner, allowing many different views of a large body of shared information to be generated from a common underlying database structure that does not predetermine what can be recorded and how it can be analyzed.

51 To do this, we must embrace a high degree of atomization in our database design. We can design data objects and the linkages among them in a way that is less atomized or more atomized, depending on the nature of the data and how it will be used. In many database applications, compromises are made to reduce the number of linkages required between distinct data objects (e.g., to reduce the number of table joins needed in a relational system). Information about a class of entities and their properties will often be embedded in a single table (e.g., data about the customers of a business enterprise). This kind of database table therefore resembles a digital document, even though the entities in the table may be linked to other entities in the database via unique keys, allowing the database to transcend the document paradigm, at least in some respects. Many commercial databases are “class-based” in this way; in other words, their table structures depend on a predetermined classification of the entities of interest. A class-based database uses two-dimensional tables to represent predefined classes of entities that have predefined common properties (i.e., predefined by the database designer). Typically, each class of entities is represented by a single table; each entity in the class is represented by a row in the table; and each property of that class of entities is represented by a column in the table.

52 In an academic domain such as the humanities, however, a class-based database is often inadequate. A higher degree of atomization is needed to permit the flexibility in description and analysis that critical scholarship demands. The alternative to a class-based database is a highly atomized “item-based” database in which not just each entity of interest but each property of an entity and each value of a property is represented as a separately addressable data object. This enables many different dynamically generated classifications of entities, which are determined by the end-users of the system and are not predetermined by the database designer. It is important to note that even though relational databases use tables, they are not necessarily class-based but can be designed to be item-based. Likewise, non-relational databases can be item-based or class-based, depending on their design.[15]

53 A highly atomized item-based database of the kind we advocate does not eliminate the need for strings and tables as a means of presenting information to end-users. Computer programmers distinguish the secondary presentation of information to human beings from the primary representation of this information within a computer. One-dimensional character strings displayed as lines of text on a computer screen are necessary for presenting information to human readers, and there is no question that a two-dimensional table is an effective way of presenting highly structured information. But one-dimensional strings and two-dimensional tables are not the best structures to use internally for the primary digital representation of scholarly information. These structures make it difficult to manage a large number of distinct but interrelated units of information — in our case, units that reflect the wide range of entities, properties, and

relationships distinguished by scholars — in a way that allows the information to be easily combined in new configurations.[16]

54 We contend that a highly atomized item-based database can provide a common solution for scholarly work both with unstructured texts and with more highly structured data, allowing the same underlying database structure and algorithms to be used for both kinds of data. In such a database, individual data objects may be linked together by end-users in different ways without imposing a single standardized terminology or classification scheme. By virtue of being highly atomized and readily reconfigurable, data objects that represent individual entities and properties and the relationships among them are able to represent the many idiosyncratic interpretations that characterize critical scholarship much better than traditional digital documents. They can do so because there is no requirement that the entities and properties of interest be defined in advance or grouped together structurally in a way that inhibits other ways of configuring them.[17]

55 With respect to texts, in particular, we can solve the problem of overlapping hierarchies by using an item-based database to separate the representation of each hierarchy of textual components from the representation of those components themselves, while also separating the representation of each textual component from the representations of all the others. If we separate the representation of a hierarchy of textual components from the representations of each individual component we obtain an atomized structure in which each textual component is stored and retrieved separately from the hierarchies in which it participates, no matter how small it may be and no matter how it may be defined. For example, a database item may represent a unit of analysis on the epigraphic level, like a character or line; or it may represent a unit of analysis on the linguistic or discourse level, like a morpheme, word, clause, or sentence. It is up to the end-user to define the scope of the items that make up a text and how they are related to one another hierarchically and non-hierarchically.

56 We call this an item-based database because it treats each textual component as an individually addressable item of information. It is also a hierarchical database, insofar as it makes full use of the expressive power of hierarchies — as many as are needed — to represent efficiently the relationships of parts to a larger whole. Indeed, each hierarchy is itself a database item. And because this kind of database is highly atomized and item-based, it readily allows non-hierarchical configurations of items, if those are necessary to express the relationships of interest. This database design has been implemented in a working system in use at the University of Chicago [Schloen and Schloen 2012], which is described below in more detail. The same database structure has proved to be applicable to many kinds of scholarly information other than texts.

57 We are aware that this way of thinking about texts as databases and not as documents will seem strange to many people. Some will no doubt react by insisting that a text is intrinsically a sequential phenomenon with a beginning, middle, and end, and is therefore well suited to digital representation by means of a linear character sequence (i.e., a document). It is true, of course, that a text is sequential. However, as we noted above, what we are representing is not the text per se but various readers' understandings of the text; and anyone who understands what he or she is reading forms a mental representation that encompasses many parts of the text at once. A human reader's way of comprehending (literally, "grasping together") the sequentially inscribed components of a text will necessarily transcend the sequence itself. Thus, we need digital data structures that preserve the text's linearity while also being able to capture the many ways scholarly readers might conceptualize the text's components and simultaneously relate them to one another.

58 This way of thinking about texts requires a leap of abstraction of a kind familiar to database designers. It is a matter of going beyond the concrete form that information may take in a particular situation in order to discern fundamental entities and relationships that a database system can work with to generate many different (and perhaps unexpected) views of the data, depending on the questions being asked, without error-prone duplication of information. Some may object that operating at this level of abstraction — abandoning the simple and intuitive document paradigm in favor of the more complex database paradigm — creates obstacles for software development. It is true that it calls for properly trained programmers rather than do-it-yourself coding, but this is hardly an argument against developing more powerful systems that can more effectively meet the needs of researchers. Scholars rely on complex professionally written

software every day for all sorts of data management and retrieval (e.g., for social media, shopping, banking, etc.). We should not be surprised if they need similarly complex software for their own research data. As end-users they do not need to see or understand the underlying software and data structures as long as the database entities, relationships, and algorithms they are working with are understood at a conceptual level. Thus the complexity of the software should be irrelevant to them.

59 Moreover, poverty is not an excuse for failing to develop suitable software for the humanities, whose practitioners usually have little or no money to hire programmers. To the extent that a database system meets common needs within a research community whose members use similar methods and materials (literary, historical, etc.), the cost of developing and maintaining the software can be spread over many researchers. Front-end user interfaces for the same system can be customized for different projects without losing economies of scale, which depend on having a single codebase for the back-end database software, where the complexity lies. Indeed, if some of the resources expended over the years on document-oriented software for the humanities had been spent on designing and building suitable back-end database software, we would have better tools at our disposal today. In our view, universities can support digital humanities in a better and more cost-effective way by discouraging the development of idiosyncratic project-specific tools and giving researchers and students access to larger shared systems that are professionally developed and maintained, together with the necessary training and technical support.

4. XML, the CHOIR ontology, and the OCHRE database system

60 We need now to give some indication of how, in practice, one can work within the database paradigm using currently available software techniques to overcome the limitations of the document paradigm. For a long time, the barrier between textual scholarship and database systems seemed insurmountable because it was cumbersome (though not impossible) to represent free-form texts and their flexible hierarchies within relational databases, which have been the dominant type of database since the 1980s. For this reason, in spite of its limitations, the document-markup approach often seemed to be the best way to handle digital texts. However, in the last ten years the database paradigm has itself been greatly enriched by the document paradigm. This has enabled the creation of non-relational database systems that can more easily handle flexible textual hierarchies as well as highly structured data.

61 This came about because of the Extensible Markup Language (XML), which was adopted as a universal data-formatting standard by the World Wide Web Consortium in 1998 and was followed in subsequent years by other widely used standards, such as XQuery, a querying language specifically designed to work with XML documents, and the Resource Description Framework (RDF) and its SPARQL querying language, which are the basis of the so-called Semantic Web.[18] XML is itself simply a streamlined and Web-oriented version of SGML that was developed to remedy the semantic ambiguity of the HyperText Markup Language (HTML) on which the Web is based. XML makes it easy to define semantic tagging schemes and share them on the Internet.[19]

62 Database specialists quickly realized that XML transcends the world of document markup and provides a formal notation for what has been called the “semistructured data model,” in distinction from Codd’s relational data model. XML accordingly became the basis of a new kind of non-relational database system that can work equally well with structured tables and more loosely structured texts.[20] It provides a bridge between the document paradigm and the database paradigm. The keyed and indexed data objects in an XML database conform to the XML markup standard and thus may be considered documents, but they function quite differently from traditional marked-up documents within the document paradigm. Like any other XML documents, the data objects in an XML database each consist of a sequence of character codes and so can be displayed as plain text, but they are not necessarily correlated one-for-one to ordinary documents or texts in the real world. Instead, they function as digital representations of potentially quite atomized pieces of interlinked information, as in any other database system.[21]

63 For our purposes, the most important feature of the semistructured data model is its close congruence to item-based ontologies characterized by recursive hierarchies of unpredictable depth, for which the relational data model is not well suited. As we have said, an item-based ontology defined at the right level of abstraction can represent overlapping textual hierarchies without duplicating any textual content, enabling digital text representations that capture different

ways of reading a text without losing sight of the fact that they are all readings of the same text. And the same ontology can represent highly structured information without embedding units of scholarly analysis in two-dimensional tables of the kind prescribed by the document paradigm. Instead, each entity of interest, and each property of an entity, can be stored as a separately addressable unit of information that can be combined with other units and presented in many different ways.

64 We have demonstrated this in practice by developing and testing an item-based ontology we have called CHOIR, which stands for “Comprehensive Hierarchical Ontology for Integrative Research.” But an ontology is not itself a working data-retrieval system; it is just a conceptual description of entities and relationships in a given domain of knowledge. To demonstrate its utility for scholarly research, we have implemented the CHOIR ontology in an XML database system called OCHRE, which stands for “Online Cultural and Historical Research Environment.” This multi-user system has been extensively tested for the past several years by twenty academic projects in different branches of textual study, archaeology, and history. In 2012 it was made more widely available via the University of Chicago’s OCHRE Data Service [http://ochre.uchicago.edu].[22]

65 Although we have implemented the CHOIR ontology in an XML database system, we are not using XML documents as a vehicle for marked-up texts in the manner prescribed by the TEI text-encoding method. In an OCHRE database, XML documents do not correspond to real-world documents but are highly atomized and interlinked data objects, analogous to the “tuples” (table rows) that are the basic structural units in a relational database. Each XML document represents an individual entity or a property of an entity, however these may be defined in a given scholarly project; or an XML document might represent a particular configuration of entities or properties in a hierarchy or set. For example, in the case of a text, each textual component, down to the level of graphemes and morphemes if desired, would be represented by a different XML document, and each hierarchical analysis of the text would be represented by its own XML document that would contain pointers to the various textual components.

66 In contrast to conventional text-encoding methods, OCHRE’s representation of a text is not limited to a single hierarchy of textual components. If overlapping hierarchies are needed to represent alternate ways of reading the same text, OCHRE does not duplicate textual content within multiple, disconnected hierarchies. Instead, the same textual components can be reused in any number of quite different hierarchical configurations, none of which is structurally primary. Moreover, an OCHRE hierarchy that represents an overall mode of textual analysis (e.g., metrical, grammatical, topical, etc.) can accommodate smaller textual or editorial variants within the same mode of analysis as overlapping branches in the same hierarchy. OCHRE also permits non-hierarchical configurations of textual components, if these are needed.

67 In terms of Hjelmslev’s expression-versus-content distinction, which was mentioned above in connection with Dino Buzzetti’s critique of embedded markup, a text is represented in OCHRE by means of recursive hierarchies of uniquely keyed text-expression items (physical epigraphic units) and separate recursive hierarchies of uniquely keyed text-content items (linguistically meaningful discourse units). There is one hierarchy for each epigraphic analysis and one hierarchy for each discourse analysis. To connect the hierarchies that make up the text, each epigraphic unit is linked in a cross-hierarchical fashion to one or more discourse units that represent interpretations of the epigraphic signs by one or more editors. Each epigraphic unit and each discourse unit is a separately addressable database item that can have its own descriptive properties and can in turn contain other epigraphic units or discourse units, respectively, allowing for recursion within each hierarchy.

68 A recursive hierarchy is one in which the same structure is repeated at successive levels of hierarchical nesting. Many linguists, following Noam Chomsky, believe that recursion is a fundamental property of human language and, indeed, is what distinguishes human language from other kinds of animal communication (see [Hauser et al. 2002]; [Nevins et al. 2009]; [Corballis 2011]). Whether or not this is true, it is clear that recursive hierarchies are easily understood by most people and serve as an intuitive mechanism for organizing information, as well as being easy to work with computationally because they lend themselves to an iterative “divide and conquer” approach. For this reason, recursion plays a major role in the CHOIR ontology and the OCHRE database system that implements it.

The units of analysis within a given recursive hierarchy are defined by the text's editor according to his or her chosen method of analysis and they are ramified to whatever level of detail is required. For example, a scholar might distinguish pages, sections, lines, and individual signs, in the case of epigraphic-expression hierarchies, and might distinguish paragraphs, sentences, clauses, phrases, words, and morphemes, in the case of discourse-content hierarchies. But note that the nature of the textual components and the degree of nesting in each hierarchy are not predetermined by the software; they are determined by the scholars who are using the software. 69

Finally, OCHRE is not dependent on existing character-encoding standards for its representation of writing systems. These standards may be inadequate for scholarly research in certain cases. As we observed at the beginning of this paper, the Unicode character-encoding standard, large as it is, does not include every allograph of every sign in every human writing system. It embodies someone else's prior choices about how to encode characters and what to leave out of the encoding. With this in mind, OCHRE safeguards the semantic authority of the individual scholar, even at this most basic level, by abstracting the digital representation of texts in a way that allows the explicit representation, not just of texts themselves, but also of the writing systems that are instantiated in the texts. If necessary, a scholar can go beyond existing standardized character codes and can represent the components and structure of a particular writing system in terms of hierarchies of idealized signs constituted by one or more allographs (e.g., the first letter in the Latin alphabet is constituted by the allographs "A" and "a" and various other ways of writing this sign). The allographs of a writing system can then be linked individually, usually in an automated fashion, to a text's epigraphic-expression units, and the text can be displayed visually via the drawings or images associated with the allographs. 70

In other words, scholars are able to make explicit in the database not just their interpretations of a text but also their understandings of the writing system used to inscribe the text. The epigraphic expression of a text can be represented as a particular configuration of individually addressable allographs that constitute signs in one or more writing systems — or, from another perspective, a writing system can be represented as a configuration of ideal signs derived from each sign's allographic instantiations in a corpus of texts. This is implemented internally by means of pointers from database items that represent the text's epigraphic units to database items that represent signs in a writing system. Moreover, overlapping hierarchies can be used to represent different understandings of a given writing system, reconfiguring the same signs in different ways, just as overlapping hierarchies represent different readings of a given text, reconfiguring the same textual components in different ways. 71

We can now say with some confidence, having developed and tested the CHOIR ontology over a period of more than ten years in consultation with a diverse group of researchers, that this ontology is an effective solution to complex problems of scholarly data representation and analysis, including the longstanding problem of overlapping hierarchies. To be sure, writing the software to implement this ontology in a working database system such as OCHRE requires a high level of programming skill and a considerable investment of time and effort. However, the generic design of the underlying ontology ensures that the same software can be used by many people for a wide range of projects, making the system financially sustainable. 72

Professionally maintained and well-supported multi-project systems like this are much too scarce in the humanities, which have tended toward idiosyncratic project-specific software that is not adequately documented, supported, or updated over time. This tendency is not simply caused by a lack of resources. At its heart, the problem is ontological, so to speak. Scholars have lacked a comprehensive abstract ontology suitable for the full range of information with which they deal, including both free-form texts and highly structured data. A comprehensive ontology is needed as the basis for any widely shared database system intended for use by humanities scholars. Moreover, widely shared systems are highly desirable — both intellectually, because they make it easy to search and analyze large amounts of information of diverse origins, and also practically, because they can be sustained and enhanced over the long term as scholars at different institutions pool their resources to support the same system, achieving economies of scale. 73

This does not mean putting all of our digital eggs in one basket. An ontology is distinct from its implementations, so scholars who share a common abstract ontology do not all have to use the same database system. If they can muster the resources to do so, they are free to create different implementations of the same ontology while still retaining interoperability, because converting data from one database schema to another is relatively easy if the different 74

database systems employ the same underlying ontology. Note the distinction here between conceptual ontologies and database schemas. A schema is an implementation of an ontology in data structures designed for a working data-retrieval system in accordance with a particular data model (e.g., the table schemas of relational databases and the XML document schemas of semistructured databases).

In the case of CHOIR, we have implemented the ontology in an XML database system based on the semistructured data model; however, it does not depend on XML. The same ontology could also be implemented (albeit less efficiently, in our view) in a system predicated on a quite different data model — for example, a system based on the well-known relational data model or one based on the graph data model currently advocated by proponents of the Semantic Web. The CHOIR ontology is, in fact, highly compatible with the basic design of the Semantic Web. 75

5. A comprehensive ontology enables interoperability in the Semantic Web of data

Obtaining the benefits of a comprehensive ontology — not just intellectually, with respect to a fuller representation of scholarly knowledge, but also practically, with respect to economies of scale, interoperability, and sustainability — will require working within the database paradigm to create multi-project database systems that have a broad base of support. In contrast, most humanities software development has been trapped in the document paradigm exemplified by the TEI markup scheme and, more generally, by the reliance on Web documents as primary repositories of scholarly information.[23] 76

However, a major effort to reshape the Web — the Semantic Web initiative of the World Wide Web Consortium — has increasingly come to the attention of people working in digital humanities [http://www.w3.org/standards/semanticweb]. The Semantic Web can be regarded as an attempt to move the Web itself from the document paradigm toward the database paradigm. It builds upon the technical standards that underlie the document-oriented Web in order to implement what are, in effect, large databases that can be easily distributed on many different computers connected via the Internet. 77

It will be useful at this point to explain why this attempt is being made, that is, what deficiency of the traditional Web the Semantic Web is meant to address. This will help us to explain why in designing our own system we have rejected traditional documents as a means of primary knowledge representation for scholars. The design of our database system predates the Semantic Web and we have implemented it using the semistructured data model and XML rather than the graph data model and RDF, but it is nonetheless compatible with the architecture of Internet-mediated knowledge dissemination currently being encouraged by Tim Berners-Lee as founder and director of the World Wide Web Consortium. In fact, our system design, which involves implementing a hierarchical item-based ontology in an atomized and keyed database, which we first did in the 1990s using a relational database, has become much easier to implement in recent years because the best features of the document paradigm have now been incorporated into the database paradigm thanks to the non-proprietary standards published by the World Wide Web Consortium — namely, XML and XQuery, as well as Semantic Web standards such as RDF, the RDF-based Web Ontology Language (OWL), and the SPARQL querying language. 78

As we have said, the Semantic Web can be viewed as an attempt to move the Web itself from the document paradigm, in which it originated, toward the database paradigm — to go beyond a “Web of documents” to a “Web of data,” as its proponents have put it. They hope to achieve this through a revival of the graph data model for database design, which had been neglected for many years in favor of the relational data model. The heart of the Semantic Web is RDF (Resource Description Framework), which prescribes a standardized way of representing knowledge in the form of subject-predicate-object “triples.” In terms of mathematical graph theory, a set of RDF triples represents a graph structure that consists of “nodes” and “arcs” (also called “vertices” and “edges”) which can be displayed visually in a network diagram and analyzed by algorithms designed to traverse graphs (for an introduction to the graph data model and graph theory, see [Aho and Ullman 1995, 451–528]). The graph data model is universal, like the relational data model, which means that the concepts and relationships of any ontology, no matter how complex, can be represented by means of atomized RDF triples within a graph database. 79

The emergence of RDF as a database format constitutes a challenge to the practice, common in the humanities, of using digital documents as primary repositories of scholarly information, as opposed to using them simply as ephemeral secondary vehicles for presenting to human readers particular views of information that have been dynamically generated from an underlying atomized database (i.e., via HTML documents or via tabular displays of structured data). Since the advent of the World Wide Web in the early 1990s, a great deal of digital information produced by scholars has been stored in the form of tagged documents and other digital resources to which such documents are linked. Encoding texts in TEI-XML documents is an example of this strategy. But the Semantic Web calls into question this way of storing scholarly data. 80

We should keep in mind that the World Wide Web was not originally designed for the primary storage of research data, even though it has come to be used that way in the humanities. The Web was designed to present information to human readers, as is indicated by its basic metaphor of “pages” of information that an end-user can “browse” and from which a user can jump to other pages via “hyperlinks.” It is not surprising, therefore, that HTML documents are difficult to search and manipulate as data objects using database techniques. Neither the meaning of the data objects nor the meaning of the linkages among them is explicitly specified. Databases with specified meanings are of course accessible on the Web via links from static Web pages or via dynamic Web pages that the databases themselves have generated, but these databases are not part of the World Wide Web itself. They have no universal predictable structure and so require special-purpose software in each case to interpret them. The Web per se, regarded as a predictably structured database based on a common standard, is extremely simple, consisting of interlinked HTML documents and other data objects whose type and location (but not contents) can be described in HTML.[24] Because HTML was originally designed to enable human browsing of information on the Internet and was not designed for automated querying, it specifies how information should be displayed but does not specify its meaning except to say that certain units of information are linked to other units in some fashion. 81

The simplicity of the Web’s schema has made it easy for people to share information on the Internet. Scholars in the humanities have benefited because the Web has encouraged the digitization and dissemination of vast quantities of primary research material — literary texts, historical archives, maps, photographs, audio and video clips, architectural and artifactual descriptions, etc. — as well as secondary literature in the form of monographs, dictionaries, and journal articles. However, the Web’s simplicity means that it lacks a predefined semantic structure that could allow scholars to disseminate detailed analyses of their sources in a way that is conducive to automated retrieval and analysis, as opposed to human browsing and reading.[25] The internal semantic structure of a Web page is not predictable and the links between pages are semantically ambiguous, thus the complex conceptual distinctions scholars employ as they study and discuss their sources are not represented in a predictable digital form. This makes it difficult to find and compare specific observations and interpretations in a comprehensive way. Human readers can supply the semantics of the information presented to them in a Web page but no semantic constraints are contained in the internal, machine-readable representation of the information, preventing automated querying and comparison. 82

This limitation results from the Web’s reliance on the document paradigm of software design, which is exemplified in the HTML markup scheme and the Web “page” concept. The Web has used the document paradigm to great effect, popularizing a simple technique for disseminating information on the Internet in the form of marked-up character sequences. But this technique by itself is not sufficient to allow the Web to be used as a database. For this reason, the World Wide Web Consortium has built upon the XML and RDF data standards by sponsoring the creation of two new querying languages analogous to the SQL language used in relational database systems. These languages are XQuery, for use with XML documents, and SPARQL, for use with RDF triples. The advent of these new querying languages and their implementation in recent years in high-performance database software (e.g., Oracle, IBM’s DB2, MarkLogic) has made it possible to create non-relational systems capable of representing complex scholarly conceptualizations in a form that can be easily disseminated on the Internet and can also be efficiently searched and manipulated in the same way as relational databases. 83

XML itself was originally designed not as a database format but to represent the semantic structure of documents in a way that could be shared on the old-fashioned Web via browsers and other document-oriented software. Similarly, RDF 84

was originally intended simply to describe Web documents and other digital “resources” so they would be easier to find — hence the name *Resource Description Framework.* But when XML documents or RDF triples are used in a database system with a powerful querying language, they transcend the document paradigm within which they emerged. Their character sequences are used within a database to represent atomized multi-dimensional structures that go beyond the one- and two-dimensional structures prescribed by the document paradigm.

The markup method is here detached from its origins as a means of representing a text via a long sequence of characters. Marked-up character sequences function simply as a convenient way to serialize complex data structures — including the atomized, interconnected, and thus multi-dimensional structures used in databases — for the purpose of transmitting them across the Internet in a standardized “cross-platform” way, without having to worry about which operating system is used by the computing devices that are receiving them. A scholarly text, in particular, will be represented in a properly designed XML database or RDF database by many different character strings that do not comprise a single sequential data object but constitute a collection of interrelated data objects. 85

We have used the XML Schema standard and the XQuery language to implement the CHOIR ontology in a multi-project, multi-user, password-protected system — an Online Cultural and Historical Research Environment — which can be distributed widely on the Internet on many different database servers. The information stored in the OCHRE system is highly atomized. It is contained in individually addressable data objects, each of which has a universally unique identifier. This is in keeping with the architectural design of the Semantic Web, even though we are using XML documents rather than RDF triples as the basic structural components.[26] 86

However, like any conceptual ontology, CHOIR is not restricted to a single implementation. It could also be implemented in an RDF system based on the graph model currently promoted by advocates of the Semantic Web — or, for that matter, it could be implemented in a relational system using SQL. The decision about how best to implement a conceptual ontology is a matter of software engineering and depends on the software tools and standards available at the time. We have chosen to use XML documents with XQuery rather than RDF triples with SPARQL because CHOIR makes extensive use of recursive hierarchies, and XQuery, unlike SPARQL, can deal with hierarchies very efficiently while still making it easy to search and update non-hierarchical networks, on the one hand, and highly structured data, on the other.[27] But what matters most in the long run is the ontology, not the implementation. Converting data from one database schema to another — from an XML schema to an RDF schema or a relational schema — is a relatively simple matter provided that the underlying ontology remains the same. Moreover, the database need not be a closed “silo” of information. Regardless of which data model and querying language are used, a properly designed item-based system can be made accessible online and the information it manages can be widely distributed on multiple servers hosted by different institutions, in the spirit of the Semantic Web. 87

We argue that computer-aided research in the humanities would be greatly enhanced by the widespread use of a comprehensive ontology of scholarly knowledge. Accordingly, we have designed the CHOIR ontology to be a stable and sustainable basis for storing and integrating diverse kinds of information over the long term even as database software and schemas change. We offer this ontology as an example of a comprehensive conceptualization of scholarly knowledge that can be implemented in quite different kinds of database systems while serving as a widely shared conceptual framework for the digital representation of scholarly information. By means of database systems and APIs (application programming interfaces) that implement the ontology, any and all scholarly observations and interpretations, no matter how idiosyncratic, can be distributed on the Internet in a predictable digital form, allowing them to be searched and edited by interoperable software applications in a highly granular fashion with the aid of powerful querying languages. And the Semantic Web standards make it easy to disseminate the underlying ontology itself because the subject-predicate-object triples of RDF are not just a data format for a particular kind of database implementation but also provide a standardized way to express and share conceptual ontologies in an implementation-independent manner. 88

The CHOIR ontology is described in considerable detail in our OCHRE manual ([Schloen and Schloen 2012], which is available as a printed book or online at http://ochre.uchicago.edu). The ontology is currently expressed as a set of interrelated XML document types, which are the basis of the OCHRE database system.[28] To complement the CHOIR- 89

XML version of the ontology, we are in the process of creating a CHOIR-RDF version, employing the Web Ontology Language (OWL) to prescribe RDF subject-predicate-object triples that express the ontology [http://www.w3.org/TR/owl2-overview]. This will provide a standardized archival and exchange format that can be used by any CHOIR-based system, regardless of its own data model and schema, because each system can use the CHOIR-RDF format to export data in a way that preserves all the atomized entities and relationships it contains, allowing them to be imported subsequently into a different system in the form of RDF triples without any loss of information.[29]

6. Data integration without forced standardization

Earlier in this article we stressed the limitations of traditional digital documents as a means of representing scholarly texts. But the vision of scholarly computing that has shaped the CHOIR ontology and has been implemented in the OCHRE database system goes far beyond representing multiple readings of an individual text. As we explain in our OCHRE manual [Schloen and Schloen 2012], large textual corpora can be densely interrelated within the OCHRE database by means of cross-cutting links from one text to another, and also by means of synthetic works like dictionaries and bibliographies, which are themselves modeled as hierarchies of entities that are cross-linked to selected textual components. For example, a dictionary entry that cites a textual excerpt to explain a meaning or sub-meaning of a word can be linked to a text in a way that avoids any repetition of textual content. In other words, the same textual components can participate, not just in multiple overlapping hierarchies of textual analysis, but also in multiple dictionary entries, providing live links from a dictionary to its underlying texts without any error-prone duplication of data. 90

And, going beyond texts, the CHOIR ontology encompasses units of space, time, and agency as distinct entities, allowing not just texts but their authors and editors, and the places and periods of textual production, to be represented as individual database items that can be interlinked in many different ways. The goal is to capture explicitly, in a reproducible digital form, any and all distinctions and relationships a scholar might detect and discuss — and not just in textual studies but also in archaeology, history, and many other cultural and social disciplines.[30] Doing so will facilitate powerful automated analyses that span many different research projects, allowing scholars to explore the interconnections among texts and their larger worlds, however these may be defined. 91

This brings us to a central issue in scholarly computing and in many other kinds of computing: namely, the need to integrate large quantities of diverse digital information that has been recorded using different software applications based on different schemas that are based in turn on different ontologies (see [Doan et al. 2012]). Scholars encode texts using heterogeneous markup schemes; they create data tables using heterogeneous table structures; and they describe digital resources using heterogeneous metadata formats. The OCHRE system was designed from the beginning to facilitate the integration of heterogeneous information, whether it be digitized originally in the form of marked-up character sequences or in the rows and columns of structured tables. OCHRE can do this because the CHOIR ontology on which it is based is a highly generic “upper” ontology. An upper (or foundation) ontology is one that has been defined on a very general level of abstraction in order to subsume more specific local ontologies — e.g., particular nomenclatures and classifications of entities — and thus to integrate data across separate domains of knowledge. A database system that implements the CHOIR ontology can integrate information from many different researchers and research projects while preserving the terminology and conceptual distinctions of each. There is no need for a universal standardized terminology because CHOIR provides a common integrative structure at the level of the fundamental spatial, temporal, linguistic, and logical entities and relationships that are used by all scholars. 92

In practice, this entails importing into a central database a diverse array of existing scholarly data in the form of heterogeneously tagged text files and heterogeneously structured data tables, automatically parsing the existing data in order to decompose each text or table into a highly atomized collection of individual items, and then populating the database with these items while creating explicit links among them that replicate the entities, properties, and relationships inherent in the original text files and data tables. This is made possible by adopting an item-based organization of information rather than a class-based organization. In a class-based database, the number and types of the entity-classes, represented as two-dimensional tables, vary from one database to the next. Even when there are similar classes in two different databases, the predefined properties of the entities in each class, represented as table 93

columns, usually do not match. Integrating such databases requires special-purpose software that is laboriously programmed to take account of the idiosyncrasies of each database. But in an item-based database that implements the CHOIR ontology, this problem is avoided because the class-based tabular structures — and sequential textual structures, for that matter — are decomposed into smaller entities with their own properties. Classes comprised of entities that have been imported from heterogeneous data sources may then be constructed secondarily, as needed, by comparing the individual properties of the entities, without requiring that each entity be pre-assigned to a group that is assumed to have the same set of properties. And because spatial, temporal, linguistic, and taxonomic relationships are represented by flexible hierarchies, the hierarchies of one project can easily be merged with the hierarchies of another, resulting in a larger database that contains the data from both projects while remaining coherently organized in a predictable fashion.

94 Most attempts at data integration in academic disciplines have involved the imposition of a standardized terminology that is intended to be adopted by all researchers in a given domain of knowledge, whether in the form of standardized table columns, standardized markup schemes (e.g., TEI), or standardized metadata attributes (e.g., Dublin Core). However, this often proves to be unworkable as scholars chafe at the restrictions the standards impose and eventually extend them or diverge from them in idiosyncratic ways. They do this, not simply because they are wedded to their own familiar nomenclatures, but because their divergent terminologies quite legitimately reflect disparate research traditions and interpretive perspectives — not to mention the different languages spoken by researchers from different countries. No standardized terminology contains every meaningful distinction that scholars may wish to make, which means that standards tend to grow over time and become ever more bloated and complex to please everyone, requiring the software that implements them to keep changing, which in turn diminishes the value of the standards as a basis for automated data integration.

95 However, a standardized terminology is not actually necessary. Integrating heterogeneous information requires a standardized way of representing the structure of knowledge — how entities are described with properties and how entities are related to other entities and properties are related to other properties — but it does not require standardization of the properties themselves. Scholars may well find it convenient on occasion to adopt a widely shared terminology, but they should not be forced to do so.

96 For this reason, the CHOIR ontology does not impose a standardized terminology but instead provides a generic framework within which different terminologies can be expressed and compared. Property names (attributes) are not built into the structure of the database as table column headings or as predefined markup tags. Property names are themselves treated as user-defined data within a more abstract structure in which taxonomic descriptors are manipulated by end-users as individual database items in their own right. Scholars can define their own descriptive property names and property values and can organize them into a meaningful hierarchy, yielding a project-specific taxonomy. There is no structural distinction between data and metadata because all information is represented in the same way, in terms of individual items that are linked to other items. Not just entities but properties of entities are user-defined database items that are linked to the entities they describe and can themselves be constructed and manipulated as individually addressable units of information.

97 In OCHRE, a hierarchy of items that represent descriptive properties and property values forms a user-defined taxonomy, which could encompass linguistic, literary, and philological properties of the kind represented in textual markup schemes like the TEI tagging scheme, as well as properties that pertain to entities other than texts, such as persons, places, periods, and things. Texts and components of texts are entities that are represented in the database as individually addressable items, like any other entities. Resources such as images are also entities that are represented as individual database items in the same way. All units of scholarly analysis — whether they be textual, spatial, temporal, personal, or taxonomic — make use of the same data structures and algorithms, which represent each unit of analysis as a distinct database item that can be described with its own properties and related to other units by means of hierarchies and non-hierarchical networks.

98 Furthermore, database users can easily specify the semantic relationships between the disparate property names and qualitative property values found in different taxonomies; that is, they can specify thesaurus relationships indicating

synonyms, antonyms, broader terms, narrower terms, and related terms. These user-created thesaurus relationships can then be saved and used in database queries to find similar items that have been described by different projects using different terms. Data imported into the database from heterogeneously tagged texts and from heterogeneously structured tables can thus be integrated automatically without abandoning the original terminology used to record the information, which is retained permanently in the properties of the database items.

99 But just as there is no universal, standardized taxonomy that applies to all projects, there is no universal, standardized thesaurus. A thesaurus is itself a work of scholarship to be credited to a particular scholar and shared with other scholars, who can decide whose thesaurus to use when querying the database or whether to devise their own. This reflects the fact that meaning depends on context. The meanings of terms and their relationships to other terms cannot be formalized once and for all and then applied in an anonymous, automated fashion without regard to the specific intellectual context in which the terms are being used, especially in disciplines characterized by a wide range of scholarly traditions and interpretive perspectives.

100 As a matter of principle, then, we do not attempt to replace the scholarly choices of individual researchers with a standardized terminology. In the OCHRE system there are no predefined table columns or descriptive markup tags or metadata attributes. There is no predefined taxonomy or thesaurus. To impose a standardized terminology would be to deprive scholars of the ability, and the responsibility, to distinguish and describe their sources in their own way — and in a way that is exposed to scrutiny by other scholars so that interpretations can be contested and discussed. Our database design is predicated on the belief that many of the semantic distinctions important to scholars cannot be adequately formalized in advance and therefore should not be built into the internal structure of a database system but should be a matter of ongoing debate. Scholarly software should be designed to facilitate this work of interpretation, not to replace it, by making it easy to construct and to share both taxonomies and taxonomy-integrating thesauruses. But no single taxonomy or thesaurus should be assumed to be universal. These are human works that are created in particular research settings and reflect the interpretive perspectives of embodied human agents, who themselves are rooted in particular historical and linguistic traditions.

101 In other words, in the world of critical scholarship, above all, to embrace digitization is not to embrace artificial intelligence as a substitute for human judgment. In his book *What Computers Can't Do*, the philosopher Hubert Dreyfus [Dreyfus 1992] discusses the dependence of meaning on context in relation to what he sees as the limitations, in principle, of artificial intelligence — or at least the "strong" version of artificial intelligence championed by its early proponents. Following Heidegger, Dreyfus argues that our understanding of the world emerges from the background of our historically situated human embodiment. This background can never be fully articulated; thus our understanding of the world (including our scholarly judgments, in this case) cannot be adequately emulated in formal symbols and algorithms of the kind used by digital computers. Steven Horst, writing on "The Computational Theory of Mind" in the *Stanford Encyclopedia of Philosophy*, summarizes his argument as follows: "Dreyfus argued that most human knowledge and competence — particularly *expert* knowledge — cannot in fact be reduced to an algorithmic procedure, and hence is not computable in the relevant technical sense. Drawing upon insights from Heidegger and existential phenomenology, Dreyfus pointed to a principled difference between the kind of cognition one might employ when learning a skill and the kind employed by the expert. . . . more often than not, argues Dreyfus, it is not possible to capture expert knowledge in an algorithm, particularly where it draws upon general background knowledge outside the problem domain" ([Horst 2009]; see also [Winograd and Flores 1986]; [Winograd 1995]; [Winograd 2006]).[31]

102 Accordingly, we have designed the OCHRE system in such a way that a scholar who is searching the database can decide whose thesaurus to employ when retrieving information from more than one project and can easily create new thesaurus relationships, if necessary. The responsibility for correlating different terms is in the hands of the person asking the question, not the person who recorded the data, and certainly not the person who wrote the software, because semantic relationships among different scholarly taxonomies cannot be established once and for all and employed without regard to the current intellectual context. This does not rule out semi-automated ontology alignment; that is, the use of machine-learning algorithms to propose thesaurus relationships to a human expert, who then makes the final decisions about how to relate one taxonomy to another. But, even so, the semantics of the database are determined by its users rather than by an anonymous semantic authority whose interpretations have been embedded in

the software. OCHRE makes it easy to scrutinize the interpretive choices of the named scholars who created the taxonomies used when entering data and the thesaurus relationships invoked when performing a query. Users of the database are not constrained by someone else's prior interpretations, nor are they dependent on an automated search engine whose algorithm for matching similar terms is hidden from them, usurping their semantic authority in a different way.[32]

103 In its approach to data integration and to the representation of information in general, the CHOIR ontology, and the OCHRE database system that implements it, conform to long-established practices that characterize critical scholarship, in which semantic authority rests with the individual scholar. In this regard, OCHRE is different from systems that conform to practices of institutional control and standardization which have emerged in commercial and governmental contexts — in which, of course, the vast majority of existing database systems and text-processing systems have been developed, with the result that most software is designed to meet the needs of bureaucratic organizations in which semantic authority is imposed from the top down. In contrast, most scholars today would agree that how one describes an object of study and how one determines whether it is similar to something else reflect a historically situated understanding of the world. There is no absolute and universal way of describing the cultural and social world, and many would argue that the same is true for the natural world. For this reason, scholarly practices have been developed to identify who said what, and when and where they said it, in order to encourage individual interpretive autonomy and to discourage anonymous and unquestioned semantic authority. This is done by crediting descriptions and interpretations to particular named scholars and by providing mechanisms for widely disseminating their work so it can be used, criticized, and perhaps replaced by other scholars.

104 Unlike many database systems whose design has been borrowed from non-academic domains, OCHRE does not try to change these longstanding scholarly practices but rather to facilitate them. It does so by means of a database which is sufficiently well structured that it can be efficiently searched but which does not impose a predefined taxonomy or hide from view the logic by which information is retrieved. Instead, database users can construct their own taxonomy (or can borrow a taxonomy created by another project) and each OCHRE end-user can decide whose thesaurus to employ when matching terms in one project's taxonomy with those in another. Interpretive choices are exposed to view, both in the description of particular items and in the retrieval of what are deemed to be similar items. Scholars who use OCHRE are not forced to conform to "the computer" but are themselves responsible for the semantics of the database, which is not an interpretive agent in its own right but merely a tool to facilitate the human work of interpretation.

105 To put it another way: ontological heterogeneity is not a vice to be suppressed but a defining virtue of critical scholarship. Digitization of scholarly information should not be used as an excuse to suppress this heterogeneity, imposing a false consensus by institutional fiat. Instead, software should be designed to make it easy for scholars to disagree with others and defend their own interpretations — or to reach agreement, as the case may be — thereby facilitating a productive conflict of interpretations. We advocate, therefore, a highly generalized upper ontology for scholarly information that does not depend on standardizing the terms and conceptual distinctions to be used within a given domain of knowledge but instead relativizes and subsumes disparate local ontologies, including the more general but still domain-specific local ontologies that are expressed in standardized text-encoding schemes like the TEI tagging scheme, in standardized table structures that represent predefined entity classes and properties, and in standardized metadata attributes.

106 Of course, even the highly abstract CHOIR ontology that underlies the OCHRE system is not timeless or absolute. Embedded in it are historically situated assumptions about how to organize knowledge. But experience has shown that this ontology, which relies on very general concepts such as space, time, and agency, and which harnesses the expressive power of recursive hierarchies, can represent a very wide range of scholarly descriptions and interpretations in many different fields of study.

Conclusion

107 At the beginning of this article we focused on a particular problem in the digital representation of scholarly texts, namely, the problem of overlapping hierarchies. It should now be apparent that this problem is simply a special case of a much

larger problem in the representation of scholarly knowledge. Overlapping hierarchies occur in many areas of research in which there is a need to represent multiple configurations of the same units of analysis, be they spatial, temporal, textual, or defined in some other way. To achieve the full benefits of digitization, scholars need to be able to record multiple interpretations of the same entities in a predictable, reproducible digital form; and they need to be able to do so without error-prone duplication of data, without severing the complex interconnections among their various interpretations, and without being forced to adopt a standardized terminology that deprives them of semantic authority and limits the diversity of conceptualizations that motivates their scholarly work in the first place.

For digital texts, in particular, this will mean abandoning the document paradigm in favor of the database paradigm. And within the database paradigm, it will mean abandoning the class-based organization of information — itself a reflection of the intuitive but overly restrictive document paradigm — in favor of a more abstract and atomized item-based organization. Only then will we obtain a digital ontology of scholarly knowledge that is not borrowed uncritically from non-scholarly domains and is sufficient for our task. 108

Acknowledgments

We are grateful to Drayton Benner, Charles Blair, Martin Mueller, and Miller Prosser for reading drafts of this article and giving us their comments. 109

Notes

[1] We are referring here to logical data structures that an application programmer manipulates by means of a high-level programming language. Some of the simplest logical structures are one-dimensional arrays that represent character strings and two-dimensional arrays that represent tables. We do not claim that programmers can dispense with these simple structures as components of more complex data structures; rather, we are emphasizing the limitations of using a one- or two-dimensional structure not just as a building block but as the overarching way of organizing information, which is what is commonly done in humanities computing.

[2] For explanations of database “normalization,” “keys,” “joins” and other technical details, see *Database Systems: The Complete Book* by Hector Garcia-Molina, Jeffrey D. Ullman, and Jennifer Widom [Garcia-Molina et al. 2009].

[3] We are referring here to atomized and normalized database systems with semantically rich schemas suitable for comprehensive querying in a finely grained manner, as opposed to document-oriented “content-management” systems. See [Garcia-Molina et al. 2009] for an explanation of the relational data model and the SQL querying language, as well as an overview of the newer semistructured (XML) data model and XQuery querying language. Either or both of these data models can now be employed when building high-performance multi-user systems using “enterprise-level” database management software such as Oracle, IBM’s DB2, and MarkLogic. A third model, the graph data model — so called because it is based on mathematical graph theory — actually predates the relational data model in its use within database systems and is currently enjoying a revival (in the 1960s and 1970s, a graph-oriented approach was used in “network” databases). For an introduction to the graph data model, see Chapter 9 in *Foundations of Computer Science* by Aho and Ullman [Aho and Ullman 1995]. The semistructured XML model is itself a type of graph data model that is particularly well-suited for hierarchically organized data, though it is by no means limited to hierarchies; however, the term “graph database” has come to be used primarily for more loosely structured non-hierarchical networks of data elements.

[4] In philosophy, ontology is the study of being and of the basic categories of being, but we are using the term in the way it has come to be used in information science, ever since it was borrowed by artificial-intelligence researchers in the 1980s. Tom Gruber, then at Stanford University, defined it this way: “A body of formally represented knowledge is based on a *conceptualization*: the objects, concepts, and other entities that are presumed to exist in some area of interest and the relationships that hold among them. . . . An *ontology* is an explicit specification of a conceptualization” [Gruber 1993]. Note that an ontology is merely a description of concepts and relationships; it is not a technical specification for any particular data-retrieval system. An ontology must be implemented in a working system by means of a “schema” that defines logical data structures that represent what is described in the ontology (e.g., the interlinked table structures of a relational system).

[5] The ASCII and Unicode character-encoding systems ubiquitous in computers today are descendants of the telegraphic coding systems that originated in the 1830s with Samuel Morse’s quite reductive encoding of the characters used in English-language texts. Morse and his successors were, of course, mainly concerned with encoding the Latin alphabetic characters used in modern American and European orthography. As a result, these were the only universally standardized electronic characters for many years, until quite recently, making it difficult to encode non-Western texts.

[6] Indeed, the 0's and 1's of individual binary digits are themselves just conventionalized human interpretations of measurable physical differences in analogue phenomena such as electrical current, or magnetic polarity, or surface depth in a pitted metallic film encased in plastic, whose depth is measured by a laser in a CD- or DVD-player. The numeric value of what we interpret as a sequence of binary digits (abbreviated by the term "bits") is then a further interpretation in terms of the base-2 positional numeral system; and the correspondence of that number to a written character involves yet another step of interpretation. For example, the eight-bit sequence 01000001 has the numeric value sixty-five, which is the standard ASCII code for the Latin capital letter A, by computing in base-2 as follows: $0\times2^7 + 1\times2^6 + 0\times2^5 + 0\times2^4 + 0\times2^3 + 0\times2^2 + 0\times2^1 + 1\times2^0 = 65$ (as we would write it in the more familiar base-10 positional system, i.e., $6\times10^1 + 5\times10^0 = 65$).

[7] As we noted above, members of the TEI Consortium are well aware of the limitations of the markup technique. The 2013 TEI *Guidelines* express it this way: "Non-nesting information poses fundamental problems for any XML-based encoding scheme, and it must be stated at the outset that no current solution combines all the desirable attributes of formal simplicity, capacity to represent all occurring or imaginable kinds of structures, [and] suitability for formal or mechanical validation" [TEI P5 2013, 629].

[8] Often it is not necessary to encode each hierarchy manually in a laborious fashion because aspects of a reader's competence can be captured in text-processing algorithms that automatically encode distinctions and relationships a human reader would see. It is irrelevant for our purposes whether analytical hierarchies are created manually or automatically.

[9] Chapter 16 of the TEI *Guidelines* [TEI P5 2013, 495–545] discusses "Linking, Segmentation, and Alignment," with a section at the end of the chapter devoted to stand-off markup. The linking of textual components within and across TEI-encoded documents is meant to be done by means of the XML "id" attribute and the standardized XML Linking Language (XLink), together with the XML Pointing Language (XPointer) and, for stand-off markup, XML Inclusions (XInclude). But very little software has ever actually implemented the XLink standard because after it was initially proposed, early in the history of XML, it was quickly supplanted by the Resource Description Framework (RDF) mechanism for expressing complex links. Likewise, XInclude has rarely been implemented, and XPointer was replaced by XPath, which forms part of the XQuery querying language. Many software tools now implement the RDF and XQuery standards to accomplish the functions for which XLink and XPointer were intended. This is an example of the shift from a document-oriented approach to a database approach because XLink, which grows out of the document model — being an extension of the simple linking mechanism found in HTML documents intended for the navigation of information by human readers — is limited by virtue of its embedding within non-atomized document structures and so did not catch on, whereas the widely used RDF standard prescribes a highly atomized format of the kind used in databases and is much better suited for automated querying.

[10] See, e.g., [Schmidt and Colomb 2009] and references to other work cited therein. See also Chapter 20.5 of the TEI *Guidelines* on "Non-XML-based Approaches."

[11] Along these lines, an interesting foray has been made in the direction of converting TEI documents into atomized RDF triples that can be queried, database-style, using the SPARQL querying language (see [Tummarello et al. 2008]). We discuss RDF databases below.

[12] This limiting assumption was discussed already in a 1988 article by David Barnard et al. and was taken up again in 1993 by Allen Renear and some TEI colleagues in an online paper entitled "Refining Our Notion of What Text Really Is: The Problem of Overlapping Hierarchies," as they reflected on the problematic inheritance they had received from SGML, of which the TEI encoding scheme was a particular application. As they said: "During the initial development of descriptive markup systems . . . each document was seen as having a single natural representation as a 'logical' hierarchy of objects, as determined by the genre of the document. What text objects might occur in a document on this view is a function of the genre or category of text that that document belonged to . . . Although representations of a particular document might differ when there was some uncertainty about the structure of the document being represented, and the specificity or granularity of a representation could vary, there was a sense that a single document structure was being encoded" [Renear et al. 1993].

[13] IBM had invested a great deal of money to develop and market a hierarchical database system called IMS (Information Management System), which was soon made obsolete by relational database systems. IBM's reluctance to capitalize on Codd's innovation allowed Larry Ellison's Oracle Corporation to grab a large share of the relational database market, which it retains to this day. IBM itself belatedly marketed a relational system called DB2, which has now largely replaced IMS, although the latter is still used in some settings.

[14] Parsing algorithms scan the linear character sequence of a marked-up text to construct a more atomized digital representation (a "document object model") in which components of the text are distinguished and explicitly interrelated, but these algorithms cannot create a more complex model than can be expressed in the original character string. Markup tags embedded in a one-dimensional character sequence can be used to represent not only a hierarchical tree structure but also two-dimensional tables, which is why XML documents have become an important way of transmitting structured data between computing devices. But a single character string, even with markup, is not designed to represent the more complex multi-dimensional configurations possible in a database — except in the trivial case in which a group of

independent database objects (e.g., relational tuples) are serialized and combined into a single omnibus string for purposes of transmission over the Internet via HTTP.

[15] The fundamental data objects in a relational database are tuples. A set of tuples (called a “relation”) is conventionally displayed as a table with one row per tuple. However, the relational data model does not require information to be embedded in tabular rows and columns in a document-oriented fashion, that is, in such a way that each table corresponds to a particular class of entities possessing common properties, with one entity per row and one property per column, mimicking the structure of human-readable documents in which all of the information about a group of entities is stored together in a single two-dimensional configuration. Relational database systems often use class-based tables as a matter of convenience because a more highly atomized design is not necessary. But such tables do not take advantage of the ability of a relational system to store and retrieve information in a more atomized and flexible manner. The class-based approach is easy and intuitive and may be adequate in non-scholarly contexts in which modes of description are standardized in advance, as happens in a business enterprise; thus tables that correspond to predefined classes of similar entities (e.g., employees, products, customers, etc.) are frequently used as primary data structures in commercial settings. But this is an application-level design choice. The data models and querying languages that underlie general-purpose database software, both relational and non-relational, do not require class-based tables but can be used to implement a highly atomized item-based database that is not structured around predefined classes of entities but treats each unit of information, no matter how small, as a separately addressable data object that can be combined with other atomic data objects in a highly flexible way.

[16] Computer programmers make a further distinction between high-level data structures, such as strings and tables, which are manipulated by application programs, and the lower-level structures by which the high-level structures are themselves implemented in a particular computer system. These lower-level structures are even more primary, in a sense, but they are invisible to application programmers, who work with standardized logical structures. We are concerned here with data structures that are primary from the point of view of a database developer or a text-processing application developer, as opposed to what is primary to developers of the underlying system software on which application programs rest.

[17] An item-based database design bears some similarity to the object-oriented approach in software development and is well suited to object-oriented programming techniques. However, an item-based design does not have to be implemented in an object-oriented or object-relational database system but can be implemented in other kinds of systems. Moreover, the technical term “object-oriented” implies more than we intend by the term “item-based” (e.g., it implies polymorphism, encapsulation, etc.). For this reason, we have adopted the term “item-based” in contrast to “class-based.”

[18] XML defines a universal tagged-text format for transmitting structured information on the Internet among diverse computing platforms and operating systems [http://www.w3.org/standards/xml]. RDF represents information in terms of subject-predicate-object “triples” [http://www.w3.org/rdf]. SPARQL is a querying language designed for retrieving information from large collections of RDF triples. XML, XQuery, RDF, and SPARQL are all non-proprietary standardized specifications published by the World Wide Web Consortium, which was originally formed to take responsibility for the HyperText Markup Language (HTML) standard on which the Web itself is based.

[19] Like HTML files, XML files are digital documents that can be displayed as human-readable text because they are encoded at the level of binary digits according to a known standard for representing textual characters — usually the Unicode UTF-8 variable-width encoding format, of which the older ASCII standard is now a subset (see http://www.unicode.org). All recent computers support the UTF-8 standard; thus XML files can be readily transmitted from one computer to another regardless of the computer’s operating system. For this reason, XML has become an important vehicle for transmitting structured information over the Internet. For example, the TEI Consortium’s markup scheme is currently implemented by means of XML documents.

[20] For an early description of the semistructured data model and how it differs from the relational model, see [Abiteboul et al. 2000]. For a more recent treatment, see [Garcia-Molina et al. 2009, 483–515].

[21] The fact that an XML database is composed of “documents” does not make it any less a database and does not erase the clear distinction between the document paradigm and the database paradigm. A document consists of a single character string whereas a database consists of many distinct data objects. An XML document’s character string can be parsed into a “document object model” and its components can be isolated and retrieved via the XPath language, but a document (or a concatenation of documents) is limited in what it can represent, in comparison to a database. XPath is a powerful document querying language but it locates the individually tagged components in a document via their relative positions and thus remains within the document paradigm. An XML database, on the other hand, consists of many different XML documents that can be queried and updated as distinct data objects, both individually and collectively, via the XQuery language, which subsumes XPath but has many other features, being analogous to the SQL language used with relational databases. XQuery enables not just searching within individual documents but also “joins” that link different documents to one another via unique keys, in a manner analogous to the database joins among tuples (table rows) that are characteristic of a relational system. Collectively, the separate keyed documents in an

XML database can, like relational tuples, represent much more complex configurations of information than would be possible in a single document. In general, a database system transcends simple document-processing because it implements an atomized data model and corresponding querying language, which are necessary for the efficient multi-dimensional representation and manipulation of information; e.g., the relational data model with SQL, the semistructured (XML) data model with XQuery, or the network-graph (RDF) data model with SPARQL.

[22] OCHRE is a transactional multi-user system implemented in an enterprise-class DBMS. It makes use of record-locking in order to meet the "ACID" requirements of atomicity, consistency, isolation, and durability. It serves both as an active database system for entering and modifying information during the data-collection phase of a project and as an archival system for preserving and viewing the project's data over the long term.

[23] We include within the document paradigm textual "databases" that may well be implemented by means of general-purpose database software but do not decompose and atomize texts in a true database fashion (i.e., in a way that reflects the distinctions scholars routinely make when studying the texts) and so do not allow flexible reconfigurations of textual components in multiple overlapping hierarchies and non-hierarchical networks. Textual databases may decompose texts into separate lines, storing each line in a separate database record, but this imposes one particular analytical structure on a text, inhibiting other ways of analyzing the same text. And within each line there still remain long character sequences that are used as the text's primary digital representation. Software is occasionally created that uses a more atomized design for particular textual corpora or for particular analytical purposes but we are not aware of any other comprehensive item-based ontology like CHOIR.

[24] From the perspective of graph theory, Web pages are nodes and Web hyperlinks are arcs, so the World Wide Web as a whole can be regarded as a vast database predicated on the graph data model. But it is difficult to query the Web effectively as a database because it has an extremely simple ontology consisting of an amorphous network of semantically undefined units of information ("pages") linked in an undefined way to other units of information. This ontology is implemented in a very simple graph schema defined by HTML in contrast to the more complex graph schemas that can be defined by sets of RDF triples.

[25] Querying by means of a Web search engine such as Google is not an adequate solution to this problem. A search engine returns a list of Web pages ranked in order of their presumed relevance to what the user wants to find. This is helpful as a rough-and-ready way to locate information but, as every Web user knows, search engines often return irrelevant information or fail to find relevant information.

[26] A distributed database architecture is possible because each XML document in the OCHRE system is an atomic data object that has its own universally unique identifier which serves as a key field for retrieving the object and for performing database joins. This is analogous to the universally unique identifier (normally a Web URI) found in each member of an RDF triple. XML documents in the OCHRE database contain identifiers that point to other documents on the same database server or on different database servers elsewhere on the Internet. The number and types of other documents to which a given document can point will depend on its own document type. Every document is an instance of one of several possible document types, which collectively prescribe the schema of the database. Each document type is defined by means of an XML Schema specification.

[27] XML systems and RDF systems both depend on mathematical graph theory rather than the set theory that underlies relational database systems, but XML systems are optimized to work with a particular kind of graph structure consisting of nodes and arcs that represent hierarchies of data elements and attributes. In addition, the XQuery language used with XML documents is "Turing complete," whereas the SPARQL language used with RDF triples is not. This means that XQuery can perform any query performed by SPARQL but not vice versa. It also means that any query or update performed by SQL, which is the standard querying language for relational databases, can be translated into XQuery. Support for the XQuery and SPARQL standards has recently been incorporated into high-performance DBMS software sold by leading vendors such as Oracle and IBM, whose database platforms are no longer purely relational but now support the semistructured-graph data model (via XML) and the network-graph data model (via RDF). Likewise, MarkLogic, a leading XML DBMS, now supports SQL for querying relational tuples and SPARQL for querying RDF triples, as well as providing a highly efficient mechanism for indexing and querying XML documents via XQuery. MarkLogic also has built-in support for Apache Hadoop, allowing it to use the MapReduce technique to distribute data and processing among many computing nodes, ensuring scalability in the face of large amounts of data and large numbers of simultaneous users.

[28] See *OCHRE* by Schloen and Schloen (2012), pp. 372–375. Previously, we called these document types the "Archaeological Markup Language," abbreviated as ArchaeoML, reflecting our initial focus on archaeological data. However, these XML document types and the upper ontology they express have proved to be applicable to a wide range of research data beyond archaeology. Moreover, the designation "markup language" was confusing for people who assumed that our XML document types merely define metadata tags for marking up existing documents and data tables rather than representing scholarly data of all kinds, including texts and tables, in a more fundamental and atomized fashion. Thus, we have adopted the name CHOIR for the ontology (*Comprehensive Hierarchical Ontology for Integrative Research*) and CHOIR-XML for the implementation of this ontology as an XML database schema in the form of a set of interrelated XML document types,

whose structure is specified via the XML Schema language.

[29] In addition to CHOIR-RDF, OCHRE has its own archival export format consisting of XML documents generated from the database that are much larger and "flatter" than the highly atomized CHOIR-XML data objects used within the database system itself. These exported XML documents also contain XSLT stylesheets to format the information as HTML for display in Web browsers. This "denormalized" archival format is intended for use with simple searching and viewing applications that dispense with a complex database querying mechanism.

[30] The original impetus for the development of the OCHRE database system and the ontology it implements was the need to manage large amounts of heterogeneous archaeological information. But it quickly became apparent that the same data structures, and thus the same software, could be used equally well for the representation of texts and other phenomena. The basic idea behind our hierarchical, item-based approach is described in an article by David Schloen entitled "Archaeological Data Models and Web Publication Using XML" [Schloen 2001]. This article is out of date in some respects — it was written before the OCHRE project was begun and before XML databases were widely available — but it explains the basic design principle that underlies the CHOIR ontology and the OCHRE system.

[31] In their 1986 book *Understanding Computers and Cognition*, the computer scientists Terry Winograd and Fernando Flores discuss the implications for software design of Heidegger's phenomenologically oriented ontology and the view of human cognition that flows from it, which stands in radical opposition to "the rationalistic orientation that pervades not only artificial intelligence and the rest of computer science, but also much of linguistics, management theory, and cognitive science" [Winograd and Flores 1986, 6]. Twenty years later, Winograd noted that work in artificial intelligence (AI) has largely moved away from the (failed) rationalistic attempt to capture human thought in a formal symbolic representation and has moved toward "statistical, embodied, and constructivist approaches" that differ markedly from the original AI paradigm and are compatible with a phenomenological "design-oriented" approach: "Rather than basing AI competence on a logical [computer-based] representation of the setting and the agent's knowledge, there is an interplay between general adaptive mechanisms and world experience, which leads over time to intelligent behaviors, often as the outcome of extensive examples and training. . . . Of course there is a major role for a rationalistic approach in creating and understanding the adaptive mechanisms. Work in statistical language understanding, neural networks, or machine learning is based on deep analysis and quantitative models of the different mechanisms and techniques that form the basis of adaptation. But the researcher is not required (or able) to explicitly represent the knowledge or rules of behavior for the intelligent system" [Winograd 2006, 1257–1258].

[32] Concerning semi-automated ontology alignment, Alon Halevy, a computer scientist who is an expert on data integration, says: "Some argue that the way to resolve semantic heterogeneity is through standard schemas. Experience has shown, however, that standards have limited success and only in domains where the incentives to agree on standards are very strong. . . .Resolving schema heterogeneity is inherently a heuristic, human-assisted process. Unless there are very strong constraints on how the two schemas you are reconciling are different from each other, one should not hope for a completely automated solution. The goal is to reduce the time it takes human experts to create a mapping between a pair of schemas, and enable them to focus on the hardest and most ambiguous parts of the mapping" ([Halevy 2005, 54–55]; see also [Doan et al. 2012, 124–127]).

Works Cited

Abiteboul et al. 2000 Abiteboul, S., Buneman, P., and Suciu, D. *Data on the Web: From Relations to Semistructured Data and XML*. Morgan Kaufmann, San Francisco (2000).

Aho and Ullman 1995 Aho, A. V. and Ullman, J. D. *Foundations of Computer Science: C Edition*. Computer Science Press, New York (1995). http://i.stanford.edu/~ullman/focs.html

Barnard et al. 1988 Barnard, D., Hayter, R., Karababa, M., Logan, G., and McFadden, J. "SGML-Based Markup for Literary Texts: Two Problems and Some Solutions", *Computers and the Humanities*, 22 (1988): 265-276.

Buzzetti 2002 Buzzetti, D. "Digital Representation and the Text Model", *New Literary History*, 33 (2002): 61-88.

Buzzetti 2009 Buzzetti, D. "Digital Editions and Text Processing". In M. Deegan and K. Sutherland (eds), *Text Editing, Print and the Digital World*, Ashgate, Farnham, England (2009), pp. 45-61.

Buzzetti and McGann 2006 Buzzetti, D. and McGann, J. "Critical Editing in a Digital Horizon". In L. Burnard, K. O'Brien O'Keeffe, and J. Unsworth (eds), *Electronic Textual Editing*, The Modern Language Association of America, New York (2006), pp. 53-73.

Codd 1970 Codd, E. F. "A Relational Model of Data for Large Shared Data Banks", *Communications of the Association for Computing Machinery*, 13 (1970): 377-387.

Corballis 2011 Corballis, M. C. *The Recursive Mind: The Origins of Human Language, Thought, and Civilization*. Princeton

University Press, Princeton, N.J. (2011).

Cummings 2007 Cummings, J. “The Text Encoding Initiative and the Study of Literature”. In S. Schreibman and R. Siemens (eds), *A Companion to Digital Literary Studies*, Blackwell, Malden, Massachusetts (2007), pp. 451-476.

Doan et al. 2012 Doan, A., Halevy, A., and Ives, Z. *Principles of Data Integration.* Elsevier, Waltham, Massachusetts (2012).

Dreyfus 1992 Dreyfus, H. L. *What Computers Can’t Do: A Critique of Artificial Reason*. MIT Press, Cambridge, Massachusetts (2nd edition; 1992).

DuCharme 1999 DuCharme, B. *XML: The Annotated Specification*. Prentice Hall, Upper Saddle River, New Jersey (1999).

Garcia-Molina et al. 2009 Garcia-Molina, H., Ullman, J. D., and Widom, J. *Database Systems: The Complete Book*. Pearson Prentice Hall, Upper Saddle River, New Jersey (2nd edition; 2009).

Goldfarb 1990 Goldfarb, C. F. *The SGML Handbook*. Clarendon Press, Oxford (1990).

Goldfarb 1996 Goldfarb, C. F. “The Roots of SGML: A Personal Reflection”, http://www.sgmlsource.com/history/roots.htm (1996).

Gruber 1993 Gruber, T. R. “A Translation Approach to Portable Ontology Specifications”, *Knowledge Acquisition*, 5 (1993): 199-220.

Halevy 2005 Halevy, A. “Why Your Data Won’t Mix”, *Queue*, 3/8 (2005): 50-58.

Hauser et al. 2002 Hauser, M. D., Chomsky, N., and Fitch, W. T. “The Faculty of Language: What Is It, Who Has It, and How Did It Evolve?”, *Science*, 298 (2002): 1569-1579.

Horst 2009 Horst, S. “The Computational Theory of Mind”, *Stanford Encyclopedia of Philosophy*, Stanford University http://plato.stanford.edu/entries/computational-mind (2009).

Huitfeldt 1995 Huitfeldt, C. “Multi-Dimensional Texts in a One-Dimensional Medium”, *Computers and the Humanities*, 28 (1994/1995): 235-241.

McGann 1991 McGann, J. J. *The Textual Condition*. Princeton University Press, Princeton, New Jersey (1991).

McGann 2004 McGann, J. “Marking Texts of Many Dimensions”. In S. Schreibman, R. Siemens, and J. Unsworth (eds), *A Companion to Digital Humanities*, Blackwell, Malden, Massachusetts (2004), pp. 198-217. http://www.digitalhumanities.org/companion

Nevins et al. 2009 Nevins, A., Pesetsky, D., and Rodrigues, C. “Evidence and Argumentation: A Reply to Everett (2009)”, *Language*, 85 (2009): 671-681.

Renear 2004 Renear, A. H. “Text Encoding”. In S. Schreibman, R. Siemens, and J. Unsworth (eds), *A Companion to Digital Humanities*, Blackwell, Malden, Massachusetts (2004), pp. 218-239. http://www.digitalhumanities.org/companion

Renear et al. 1993 Renear, A., Mylonas, E., and Durand, D. “Refining Our Notion of What Text Really Is: The Problem of Overlapping Hierarchies”, Brown University http://www.stg.brown.edu/resources/stg/monographs/ohco.html (1993).

Schloen 2001 Schloen, J. D. “Archaeological Data Models and Web Publication Using XML”, *Computers and the Humanities*, 35 (2001): 123-152.

Schloen and Schloen 2012 Schloen, J. D. and Schloen, S. R. *OCHRE: An Online Cultural and Historical Research Environment*. Eisenbrauns, Winona Lake, Indiana (2012). http://ochre.uchicago.edu

Schmidt and Colomb 2009 Schmidt, D. and Colomb, R. “A Data Structure for Representing Multi-version Texts Online”, *International Journal of Human-Computer Studies*, 67 (2009): 497-514.

TEI P5 2013 Burnard, L. and Bauman, S. (eds), *TEI P5: Guidelines for Electronic Text Encoding and Interchange*, Text Encoding Initiative Consortium, Charlottesville, Virginia (2013). http://www.tei-c.org/release/doc/tei-p5-doc/en/html

Tummarello et al. 2008 Tummarello, G., Morbidoni, C., Puliti, P., and Piazza, F. “A Proposal for Text Encoding Based on Semantic Web Tools”, *Online Information Review*, 32 (2008): 467-477.

Winograd 1995 Winograd, T. “Heidegger and the Design of Computer Systems”. In A. Feenberg and A. Hannay (eds), *Technology and the Politics of Knowledge*, Indiana University Press, Bloomington, Indiana (1995), pp. 108-127.

Winograd 2006 Winograd, T. “Shifting Viewpoints: Artificial Intelligence and Human-Computer Interaction”. *Artificial Intelligence* 170 (2006): 1256–1258.

Winograd and Flores 1986 Winograd, T. and Flores, F. *Understanding Computers and Cognition: A New Foundation for Design*. Ablex Publishing, Norwood, New Jersey (1986).