Christopher W. Blackwell and Neel Smith

The CITE Architecture: a Conceptual and Practical Overview

Abstract: CITE, originally developed for the Homer Multitext, is a digital library architecture for identification, retrieval, manipulation, and integration of data by means of machine-actionable canonical citation. CITE stands for "Collections, Indices, Texts, and Extensions", and the acronym invokes the long history of citation as the basis for scholarly publication. Each of the four parts of CITE is based on abstract data models. Two parallel standards for citation identify data that implement those models: the CTS URN, for identifying texts and passages of text, and the CITE2 URN for identifying other data. Both of these URN citation schemes capture the necessary semantics of the data they identify, in context. In this paper we will describe the theoretical foundations of CITE, explain CTS and CITE2 URNs, describe the current state of the models for scholarly data that CITE defines, and introduce the current data formats, code libraries, utilities, and end-user applications that implement CITE.

Introduction

The very articulate Astronomer Royal Martin Rees has described the goal of science in this way:

> The aim of science is to unify disparate ideas, so we don't need to remember them all. I mean we don't need to recognize the fall of every apple, because Newton told us they all fall the same way.[1]

This remark captures a quintessential difference between the natural sciences and the humanities. Humanists, like scientists, unify disparate ideas, but we must record each unique phenomenon that we study. If we develop a unified view of ancient Greek poetry, for example, we will never conclude that "Because I am familiar with the *Iliad*, I do not have to remember the *Odyssey*," or "I have studied Greek poetry so I do not need to know about the tradition of

1 (Tippett and Rees 2013).

Christopher W. Blackwell, Furman University
Neel Smith, College of the Holy Cross

https://doi.org/10.1515/9783110599572-006

Serbo-Croatian epic." Humanists care about apples generically because of their marvelous, specific, variety.

Christopher Blackwell and Neel Smith, as Project Architects of the Homer Multitext project (HMT), originally developed the CITE[2] architecture to meet the needs of that project.[3] Together with the Editors, Casey Dué and Mary Ebbott, we recognized the need for an architecture that would outlive specific, rapidly changing technologies, while at the same time thoroughly capturing the semantics of our work in a format that both humans and machines could work with.

We were surprised to find that, despite the centuries-long tradition in disciplines like classical studies of citing texts by canonical reference, this experience had not been generalized in the digital humanities community. Even the most forward-looking digital projects a decade ago were relying on textual references that failed to represent the semantics implicit in conventional canonical citation, and were instead expressed in notations such as URLs that, while machine-actionable today, were closely tied to specific ephemeral technologies. We began work on the Canonical Text Services protocol (CTS), and eventually devised the CTS URN notation for citing texts. We subsequently applied this scheme – URN notation for citation, a service for retrieval of material identified by URN, and client software that talks to the service – to all the material in the HMT project: texts, physical artifacts like manuscripts, documentary objects like photographs, and analytical objects such as morphological analyses and syntactical graphs of texts.

CITE allows us to name the things we are studying in a very precise and flexible way. We can identify "Book 2 of the *Iliad* in any version," or "The third letter *iota* in the Greek text of *Iliad* Book 1, line 1, as it appears on Manuscript *Marcianus Graecus* Z.454 [=822]". We can identify a physical page of a manuscript as a physical page, as easily as we can identify an image of that page, and we can easily associate any number of images with a single physical artifact. We can identify smaller regions of an image with citations that can identify the part of an image that depicts a single character, while retaining the context of the larger image. With CITE we can cite abstractions as easily as concrete objects. For example, we can use CTS URNs to identify a passage of text in an edition; this is concrete data. But two readers might disagree on the syntax of that passage; that is,

2 CITE stands for "Collections, Indices, Texts, and Extensions".

3 The Homer Multitext (http://www.homermultitext.org) is a project of the Center for Hellenic Studies of Harvard University (last access 2019.01.31). It aims to document the history, tradition, and language of Greek epic poetry. Casey Dué and Mary Ebbott are its editors; Christopher Blackwell and Neel Smith are its architects.

these readers might assert two competing graphs, abstract data-objects that organize the (concrete and agreed upon) text differently. In the CITE architecture, we can work with the concrete text (identified by a CTS URN) and both of those abstract graph-objects, identified by CITE2 URNs.

CITE identifiers can capture and align versioned collections of data. For example, in the *HMT* project, we cite manuscript folios, but we do not have much to say about the physical objects beyond the fact that certain lines of the *Iliad* and certain commentary texts appear on a given folio. So our citation to "MS A, folio 12-recto [HMT edition]" points to a data record that has relatively little information: This is the 24th folio-side; it is a "recto". A codicologist might make a collection of data about this manuscript in which each object has much more data: sequence number, recto/verso, degree of gelatinization, repairs, quality of ink, etc. That collection would be "MS A, folio 12-recto [Codicology edition]". The structure of CITE URN citations allows us to have these two collections, each recording different data, but not losing the fact that "MS A, folio 12-recto" is in fact the same thing in both. Thus, in a CITE environment, a machine or human can discover "everything anyone says about MS A, folio 12-recto."

CITE is a framework independent of any particular technology. The principles of CITE would work on paper and ink as easily as in a digital computer. Its principles can be implemented in different languages, for different hardware and software. Since 2001, Blackwell and Smith have implemented CITE in Perl, XSLT, Java, Groovy, Javascript (now ECMAScript), using data stored in SQL Databases, Google BigTable, eXist XML Databases, and Fuseki RDF databases.

As of 2018, the reference implementation of CITE consists of specific libraries of code (written in the Scala language[4]). These are dedicated to specific tasks: one library is for creating and manipulating URN citations; one is for working with passages of text and textual corpora; one is for objects in collections. The "tier 1" libraries give us control over specific objects of study, "scholarly primitives". When these primitives are citable in a way that machines can work with, the "tier 2" libraries allow composition and analysis of those objects: additional code libraries are concerned with relations among objects, or more specific compositions, such as the three-way relationship among a "text-bearing artifact" (*e.g.* and inscription), a digital transcription of the text, and documentary evidence (a digital photograph).

Finally, there is CEX, the CITE Exchange format. This is a way to capture complex digital library content in a flexible, plain-text format.[5] CEX can capture

4 https://www.scala-lang.org (last access 2019.01.31).

5 https://cite-architecture.github.io/citedx/CEX-spec-3.0.1/ (last access 2019.01.31).

texts, collections of objects, and relations among citable resources. It serves large projects and small ones. A single CEX file might contain a Greek text and English translation of a single poem. But the entire Homer Multitext dataset is currently published as a single CEX file of 13.5 megabytes.

Working with texts: OHCO2, CTS URNs

The CITE Architecture evolved from initial work that was concerned with organizing editions and translations of the Homeric *Iliad* for the Homer Multitext. "Canonical Text Services" (CTS) is the set of specifications and libraries in CITE for working with texts.

CTS is based on an abstract model of "text"; it makes sense and works only in terms of that abstract model. This model defines a text as "An ordered hierarchy of citable objects."[6] It is called "OHCO2", with the '2' distinguishing it from an earlier proposed definition of text as "an ordered hierarchy of *content* objects."[7]

Citable texts are modeled as a set of citable nodes, each with four properties:

1. Each node belongs to a work hierarchy.
2. Each node is uniquely identified in a citation hierarchy.
3. Nodes are ordered within a single text.
4. Nodes may have richly structured textual content.

The CTS URN captures both the work hierarchy and the citation hierarchy.[8] It is a standard for machine-actionable canonical citation.

The work hierarchy represents texts as they are cited by scholars. Conceptually, the work hierarchy partially overlaps with the Functional Requirements for Bibliographic Records (FRBR),[9] but since FRBR aims to model bibliographic entries as they are cataloged by librarians, there are also noteworthy differences. The roof of the work hierarchy identifies any group of texts that are conventionally cited together in the naming authority's tradition. Examples could be based on concepts such as "author" (e.g., the works of Mark Twain), "geographic origin" (e.g., papyri from Oxyrhynchus), "subject

6 (Smith and Weaver 2009).

7 (DeRose et al. 1990).

8 The formal specification for CTS URNs is at http://cite-architecture.github.io/ctsurn_spec/ (last access 2019.01.31).

9 For FRBR, see the publications listed by the International Federation of Library Associations https://www.ifla.org (last access 2019.01.31).

matter" (e.g., Latin curse tablets), or any other grouping (e.g., a group of texts named the "Federalist Papers").

A CTS URN begins with namespace declarations, followed by the text-group identifier. This identifier may be followed by an identifier for a specific notional work within that group, corresponding to the work level of FRBR. This in turn may be followed with an identifier for a specific version of that work, either a translation or an edition, corresponding to the expression level of FRBR. A version identifier may be followed by an identifier for a specific exemplar of the version, corresponding to the item level of FRBR. (Note that there is no level of a CTS URN corresponding to the FRBR "manifestation.")

The passage component is a hierarchy of one or more levels expressing a logical citation scheme applying to all versions of a text. A poem might be cited by the single unit of "poetic line." A prose work might be cited by a hierarchy such as "book/chapter/section/subsection." Passage references at any level of the text's citation hierarchy may identify either a single citable node or a range indicated by the first and last nodes of the range.

If the work component of the CTS URN is at the version or exemplar level, reference to a single citable node may be extended with indexed occurrences of a substring or a range of substrings; in a reference to a range of nodes, either or both of the first and last nodes may be extended in the same way. Indexed substring references are permitted only with URNs at the version or exemplar level because they are inherently language-specific.

CTS URNs by example

`urn:cts:greekLit:tlg0012.tlg001.msA:10.1` This CTS URN has five fields, separated by a colon. The first three are namespace declarations: `urn:cts:greekLit:`, declaring that it is a URN according to the CTS specification, and that any subsequent values are guaranteed to be unique within the `greekLit` namespace.[10] The fourth field is the work hiearchy. `tlg0012` is an essentially arbitrary identifier defined, in the `greekLit` namespace, as referring to "Homer Epic". `tlg0012.tlg001` is the arbitrary identifier for "Homeric Epic, *Iliad*". `msA` identifies a specific edition of the *Iliad*, the Homer Multitext's diplomatic

10 `greekLit` is a namespace controlled by the Center for Hellenic Studies of Harvard University's "First Thousand Years of Greek" project.

transcription of the poetic text of the Venetus A manuscript (*Marcianus Graecus* Z.454 [=822]). The fifth and final field is the citation hierarchy. This URN identifies Book 10, line 1, of that particular version of the Homeric *Iliad*.

`urn:cts:greekLit:tlg0012.tlg001:10.1` In this CTS URN, there is no version identifier specified. This URN refers to every passage identified as "10.1" in any version of the *Iliad*, in any medium and in any language. Some versions of the *Iliad* do not have a passage "10.1". For example, the Bankes Papyrus in the British Library (BM Papyrus 114) contains only some verses from *Iliad* Book 24; this papyrus, then, is not included in the texts this URN identifies.

`urn:cts:greekLit:tlg0012.tlg001.villoison:10.1` This URN identifies Book 10, line 1, in the print edition published by Jean-Baptiste-Gaspard d'Ansse de Villoison in 1788. CTS URNs are not limited to identifying digital texts.

`urn:cts:greekLit:tlg0012.tlg001.villoison.tj4265:10.1` The work hierarchy of this CTS URN has an additional record after the version identifier. This identifies an *exemplar*, a specific instance of a version of the text. In this case, the URN identifies Book 10, line 1 in Thomas Jefferson's personal copy of Villoison's 1788 edition of the Homeric *Iliad*.[11]

In any CTS URN, the citation component is optional.

`urn:cts:greekLit:tlg0012.tlg001.villoison.tj4265:` identifies Jefferson's copy of this edition in its entirety (note the final colon, required by the specification).

`urn:cts:greekLit:tlg0012.tlg001:` accordingly refers to the *Iliad* in general, any and all versions of it.

Analytical exemplars

With physical books, an exemplar is a specific copy, such as Thomas Jefferson's personal copy of Villoison's edition of the *Iliad*, mentioned above. In the digital realm the CTS definition of "exemplar" is "a text derived from an identified version according to some defined analytical process." The Homer Multitext has published a diplomatic edition of the Iliadic text of the Venetus A, identified as `urn:cts:greekLit:tlg0012.tlg001.msA:`. The project also plans to publish

11 (d'Ansse de Villoison 1788). See *The Papers of Thomas Jefferson: Volume 28 1 January 1794 to 29 February 1796* (Princeton University: 2000) index: https://jeffersonpapers.princeton.edu/alpha-glossary/64/v (last access 2019.01.31).

a transformation of that digital edition with all abbreviations expanded and the Byzantine orthography normalized to the modern orthography for ancient Greek. This derivation would be an exemplar, identified by the URN: `urn:cts:greekLit:tlg0012.tlg001.msA.normal:`.

An exemplar may also extend the citation hierarchy of the version from which it is derived. This creates a *citable tokenization*. For many kinds of analysis, it is necessary to address parts of the Iliadic text more specifically than Book + Line, tokenizing the text. An exemplar might be a specific tokenization of a version. If we were to tokenize the *Iliad* in the service of syntactic analysis, we might create an exemplar where each lexical word has a unique citation: `urn:cts:greekLit:tlg0012.tlg001.msA.syntax-tokens:1.1.1` would identify Book 1, Line 1, *token 1* of an exemplar derived from the HMT's diplomatic edition of the *Iliad*; in this tokenization, the first token (1.1.1) would be μῆνιν, the first word of the poem. 1.1.2 would be "ἄειδε", the second word.

This allows multiple, independent analyses of a version of the text to coexist. A metrical analysis of the *Iliad* might result in a citable text, of which `urn:cts:greekLit:tlg0012.tlg001.msA.metrical-feet:1.1.1` would identify the text: "μηνιν α", the first metrical foot of the *Iliad*. Note that in this exemplar, the editors might omit diacritical marks as unnecessary for this particular analysis. Both a "syntax token" and "metrical foot" exemplar can exists, uniquely an unambiguously citable, offering text-content suited to specific kinds of analysis, explicitly aligned to the edition from which they were derived, and thus implicitly aligned to each other.

Digital humanities projects have long offered tools for transforming texts. The CTS hierarchy, expressed in the CTS URN, allows us to turn those analytical transformations from *procedural methods* to *declarative objects of study* by making them subject to specific citation.

The contents of CTS texts

CTS, following the OHCO2 model, sees a "text" as, essentially, the ordered list of unique citations; the textual-content of each citation can be plain-text or text and markup of any kind. CTS is entirely agnostic of matters of language, formatting, or markup of texts. The CITE Architecture provides a mechanism for "discoverable data models", described below, which is the means by which a project can identify for automated processes, applications, or services any specifics about the text contents of a particular CTS version or exemplar.

Canonical citation vs. traditional citation

CTS URNs provide machine-actionable canonical citations that capture the semantics of a text according to the OHCO2 model. It is important to emphasize that *canonical* citation is not, here, synonymous with *traditional* citation. Canonical, here, means "unique and persistent". For some texts, the traditional scheme of citation translates well to OCHO2: the New Testament's chapter/verse, poetic line for epic poetry, book/section/subsection for the Greek historians. For other texts, the traditional scheme of citation will not work for canonical citation according to OHCO2 and CTS. The works of Plato and Aristotle, for example, traditionally cited according to pages of specific early printed editions, require an editor to define and apply a different scheme of citation. More modern works often have no citation scheme beyond "chapter" and pages in specific editions. For these, a digital editor interested in using CTS must assert a new citation scheme, such as chapter/paragraph.[12]

Working with objects: CITE Collections and CITE2 URNs

In the CITE2 model, citable objects are modeled as unique objects in versioned collections. A version of a collection is defined by its properties and their values; a versioned collection is a list of citable object properties. The CITE2 URN captures these semantics.

The values of properties in a CITE Collection are typed, but the possible types are constrained to:

- `StringType`
- `NumberType`
- `BooleanType`
- `CtsUrnType`
- `Cite2UrnType`

12 For an example of modern texts implemented as CTS texts, and published via CEX, see the CTS implementation of the novels of Jane Austen published at https://github.com/cite-architecture/citedx (last access 2019.01.31). For these, the traditional citation scheme of novel/chapter is extended by the editorial assertion of the paragraph as the leaf-node.

Properties of `StringType` can, optionally, specify a controlled vocabulary. A collection of manuscript folios, for example, might have a `side` property of type `StringType`, but constrained to values of either "recto" or "verso".

The type of a property value, and in the case of `StringType` with a controlled vocabulary, is enforced by the CITE code libraries, which will throw an exception and refuse to build a CITE Collection object with invalid data.

As an example of a Cite Collection, we represent a papyrus fragment as a collection of text-bearing surfaces. The notional collection's URN is: `urn:cite2:fufolio:poxy2099:`. In this URN, `fufolio` is a namespace, and `poxy2099` is the collection's identifier. To create a real collection, we create a citable version of this notional collection:

```
urn:cite2:fufolio:poxy2099.v1:.
```

This versioned collection has three properties: `sequence`, `rv`, `label`. Each of these is citable by URN:

```
urn:cite2:fufolio:poxy2099.v1.sequence:
urn:cite2:fufolio:poxy2099.v1.rv:
urn:cite2:fufolio:poxy2099.v1.label:
```

There are only two objects in this version of this collection:

```
urn:cite2:fufolio:poxy2099.v1:f1
urn:cite2:fufolio:poxy2099.v1:f2
```

`f1` and `f2` are arbitrary identifiers. Each of the above URNs identifies an object in the versioned collection, that is, each URN identifies all of the properties or an object with their values. Each property of an object is uniquely citable:

```
urn:cite2:fufolio:poxy2099.v1.sequence:f1
urn:cite2:fufolio:poxy2099.v1.rv:f1
urn:cite2:fufolio:poxy2099.v1.label:f1
```

Distinct objects may have identical contents, but within a collection each object is uniquely identified. Object f1 in Version v1 of this collection might have these citable property values:

```
urn:cite2:fufolio:poxy2099.v1.sequence:f1=1
urn:cite2:fufolio:poxy2099.v1.rv:f1="recto"
urn:cite2:fufolio:poxy2099.v1.label:f1="Papyrus POxy 2099, recto"
```

A collection may be referred either in the abstract as a notional collection, or concretely as a specific version of a notional collection. Each version of a collection defines a set of properties which may or may not be identical across versions, but apply to all objects within a given version. For this reason, individual objects may be canonically cited either as part of a notional or concrete collection, but individual properties can only be cited as part of a specific version of a collection.

We might have a Collection of geographical places mentioned in Herodotus: `urn:cite2:fufolio:hdtPlaces:`. We could cite one of its members with `urn:cite2:fufolio:hdtPlaces:1`. To attach actual data to this citation, we need a version of the Collection, which is defined by its properties. A very basic version of the collection might have only two properties for each object, a label and a citation to one passage of Herodotus that mentions the place:

```
urn:cite2:fufolio:hdtPlaces.v1.label:1="Halicarnassus"
urn:cite2:fufolio:hdtPlaces.v1.attestation:1=urn:cts:greekLit:tlg0016.
tlg001:1.0
```

Another version of the collection might offer richer data, or even different values for the same named property:

```
urn:cite2:fufolio:hdtPlaces.v2.label:1="Halikarnassos"
urn:cite2:fufolio:hdtPlaces.v2.attestation:1=urn:cts:greekLit:tlg0016.
tlg001:1.0
urn:cite2:fufolio:hdtPlaces.v2.pleiadesId:1="599636"
urn:cite2:fufolio:hdtPlaces.v2.latlong:1 = "37.0382205, 27.423765"
```

Here, object `1` in `v2` of this collection records a different spelling for the `label` property,[13] and adds to additional properties. The specific property values for each version can be addressed by their specific URNs, while the notional URN `urn:cite2:fufolio:hdtPlaces:1` identifies, and could be resolved to, all the values associated with that object in any version of the collection.

Collections may or may not be intrinsically ordered. The relation of citable objects in an ordered collection is analogous to the relation of citable passages in a citable text: it is possible to make statements about ordered relations at the notional level, but the ordering of citable units in individual versions are not

13 CITE is an exercise in separation of concerns, beginning with the important distinction between a *label* and an *identifier*. In our experience, it is always a mistake to try to conflate the functions of the two.

guaranteed to agree with a notional ordering. For example, in the same way that lines of a Greek tragedy might appear in a different order in different versions of the text, pages of a manuscript might have different orderings in a version recording the current bound form of a codex and a version reconstructing a different, original page sequence.

Compositions of scholarly primitives I: CITE relations

The foundation of CITE are these two categories of primitives – OHCO2 texts, and objects in collections – and the corresponding two types of URN citations that capture their semantics, CTS URNs and CITE2 URNs. This is a solid basis for documenting more complex structures as *compositions* of those primitives.

The most straightforward compositions are CITE Relations. (These are the "I" in "CITE", the "indices".) A CITE Relation has three parts: a subject, a relation, and an object.[14] Each of the three is expressed as a URN. The Subject and Object may be a CITE2 URN or a CTS URN. The Relation is a CITE2 URN, identifying an object in a collection of relation-types (or "verbs"), whose contents may be specific to a dataset or broadly applicable.

The Homer Multitext includes a collection `urn:cite2:hmt:verbs.v1:`, some of whose members include:

- `urn:cite2:hmt:verbs.v1:appearsIn` Identifying the relationship of a named person (a CITE2 URN, the subject of a relation) and the passage of the *Iliad* that mentions that person (a CTS URN, the object of the relation).
- `urn:cite2:hmt:verbs.v1:commentsOn` Identifying the relationship of a commentary text (a CTS URN, the subject of the relation) and a passage of the *Iliad* that it comments on (a CTS URN, the object of the relation).

Both of these types of relations, a character named in the text or a text that comments on another text, are potentially many-to-many relations. A passage of text might mention several characters, and a character will appear in many passage of text. Documenting these many-to-many relations is

14 CITE Relations are semantically identical to RDF Triples, and can easily be expressed as such: "Resource Description Framework (RDF): Concepts and Abstract Syntax": https://www.w3.org/TR/rdf-concepts/ (last access 2019.01.31).

simply a matter of multiplying the CITE Relations triples. So in the *HMT* 2018e data release, Achilles (`urn:cite2:hmt:pers.v1:pers1`) is mentioned in (`urn:cite2:hmt:verbs.v1:appearsIn`) 217 passages of the scholia. These 217 relations can be expressed like:

```
urn:cite2:hmt:pers.v1:pers1 # urn:cite2:hmt:verbs.v1:appearsIn # urn:
cts:greekLit:tlg5026.msA.dipl:13.A47.comment
urn:cts:greekLit:tlg5026.msA.dipl:22.36.comment
urn:cite2:hmt:pers.v1:pers1 # urn:cite2:hmt:verbs.v1:appearsIn # urn:
cts:greekLit:tlg5026.msA.dipl:13.A47.comment
...
```

By insisting that each of the three components of a relation be URNs, a body of relations can be filtered or queried according to all of the semantics captured by those URNs: all persons appearing mentioned in the intra-marginal scholia of MS A of the *Iliad*, or in Book 9 of any version of the *Iliad*; all intra-linear comments on Book 2 of the *Iliad*; all main-scholia comments on *Iliad* 1.1–1.25; etc.

Compositions of scholarly primitives II: CITE extensions

The 'E' in CITE is "Extensions", additional discoverable information providing richer composition and description of the basic scholarly primitives.

Extensions I: categorizing collections

A CITE Collection can describe a collection of images. A very basic image collection might have the properties `label`, `license`, and `caption`. (Obviously, these are collections of metadata about images, expressed as plain text; we will address actual binary image data below.) In a library where there are several different collections of images, we can distinguish them as a special category by defining an Extension. This is nothing more than another CITE Collection.

If in a library there are three collections of images:

1. `urn:cite2:hmt:venAimg.v1:`
2. `urn:cite2:hmt:venBimg.v1:`
3. `urn:cite2:hmt:e3img.v1:`

We can formally identify these three collections as belonging to a certain type by asserting a *data model* in a collection of Data Models: `urn:cite2:cite:datamodels.v1:imagemodel`, and associating each of the three image collections with that data model.

The data model itself is documented in human-readable prose online; its definition includes a link to documentation. Any user or application that is aware of the `imagemodel` data model can discover which collections in a library implement that datamodel, and (in this case) know that these collections will include at least a `label`, `license`, and `caption` property.

A user or application can ignore this association, and those collections will behave as generic CITE Collections.

Extensions II: connecting to the physical world

With collections of images in CITE, we can serialize metadata for images easily, since it is plain-text in CEX. Resolving a URN to binary image data – so the user can actually see an image – requires a connection to the physical world. A notional "image" might be resolved to a JPG file, to data delivered by the IIIF API, to a DeepZoom file, or to any combination of these.

CITE handles this by means of another "discoverable data model", additional data (itself expressed as generic CITE collections) that can identify specific collections of images as being served by one or more binary image services. By associating a CITE Collection of Images with a `binaryimg` data model, we can then publish the information necessary to resolve the image specified by URN in a CITE Collection with one or more methods for resolving that URN to a digital image:

- A type of image service (JPG file, IIIF-API, DeepZoom).
- A URL to a service hosting images from the collection.
- Filepath information necessary to resolve an image's URN to files on the server.

A working example of this is the Homer Multitext's interactive web-application.[15] The CEX of the HMT's data release identifies image collections as being exposed both as DeepZoom files and via the IIIF-API.[16] The web-application takes advantage of both of these to provide thumbnail views and interactive zooming views.

15 http://www.homermultitext.org/hmt-digital/?urn=urn:cite2:hmt:vaimg.2017a:VA304VN_0806 (last access 2019.01.31).

16 https://github.com/homermultitext/hmt-archive/blob/master/releases-cex/hmt-2018e.cex (last access 2019.01.31).

Extensions III: extension-specific predicates to URNs

In the CITE architecture we can identify passages of text at the "leaf node" level, and the CTS URN provides access to the larger context – "New Testament, John, Chapter 3, verse 16" expressed as a URN identifies a particular passage of text, but provides access to "Chapter 3" as well, or the whole "Gospel According to John", and the whole "New Testament". A CITE2 URN, likewise, can identify the value of a particular property in a particular object, or that object generically, or all objects in a particular collection. This is sound citation-practice: identifying the specific object of study in its context.

For certain kinds of data, the relationship between "object of study" and "context" requires a specifically defined notation. So a defined data model can document a model-specific *URN extension*.

In the case of the CITE `binaryimg` data model, a defined URN extension can identify a rectangular region-of-interest (ROI) on the image. The format is `URN@left,top,width,height`. A URN identifying an image of Folio 12-recto of the Venetus A manuscript is `urn:cite2:hmt:vaimg.2017a:VA012RN_0013`. To identify the ROI on that image that includes *Iliad* 1.5, we extend the URN with top, left, width, and height values, expressed as percentages of the whole image:

```
urn:cite2:hmt:vaimg.2017a:VA012RN_0013@0.1619,0.3112,0.3345,0.02451
```

This ability to extend a CITE2 URN for a specific type of object was a key to the early development of the CITE Architecture, and is the basis for the DSE Model that has become the focus of the data published by the Homer Multitext.

Extensions IV: defined compositions

DSE stands for "Documented Scholarly Editions". It is a defined data-model that can be expressed as a CITE Collection with the following properties:

- `urn` The identifiers for a DSE `Object` `(Cite2UrnType)`
- `label` A human-readable label `(StringType)`
- `text` A passage of text `(CtsUrnType)`
- `surface` A physical artifact that has the text on it `(Cite2UrnType)`
- `image` A ROI on a citable digital image `(Cite2UrnType)`

This implements a collection of citable objects, each consisting of a text, the physical artifact on which the text appears, and specific documentary evidence

that a scholar can access to see the text as it appears on the artifact. The text, artifact, and image-evidence are each individually subject to citation. But the graph that associates them is also uniquely citable.

By virtue of the CITE URNs, for each vertex in each DSE object, we have access to the larger context. One DSE Object (that is, a single 3-way graph) from the Homer Multitext is:

- URN = `urn:cite2:hmt:va_dse.v1:il10`
- Label = "`urn:cite2:hmt:va_dse.v1:il10`"
- Text = `urn:cts:greekLit:tlg0012.tlg001.msA:1.1`
- Surface = `urn:cite2:hmt:msA.v1:12r`
- Image = `urn:cite2:hmt:vaimg.2017a:VA012RN_0013.v1@0.0611,0.225,0.467,0.09`

This object, identified as `urn:cite2:hmt:va_dse.v1:il10`, is the three-way association of *Iliad* 1.1 (as it appears on the Venetus A manuscript), with folio 12 recto of the Venetus A manuscript, as evinced by image `VA012RN_0013` in version `2017a` of the collection `urn:cite2:hmt:vaimg:`, specifically in the rectangle starting at 6.11% from the top of that image, 22.5% from the left, extending to 46.7% of its width, and 9% of its height.

Extensions V: different expressions of textual data

An object in a version of a collection might have a property of type `StringType`, and that is easily discoverable with the basic CITE tools. But of course, a `StringType` might be plain text, Markdown, some form of XML, or some other encoding. It is easy to imagine a project publishing a version of a collection of comments as plain-text, and subsequently publishing a new version that adds some markup to those comments.

Because the CITE2 URN allows identification of notional collections, versioned collections, individual properties in versioned collections, in each case across the collection or filtered by an object's identifier, we can expose additional information about the nature of a property of type `StringType`.

By means of a discoverable data model, just as we associated whole collections of images with different binary image services, we can associate properties with different encodings, without losing scholarly identity.

A CITE microservice (about which see below) at http://folio2.furman.edu/lex/collections serves a transformation of the Liddell, Scott, Jones Greek Lexicon

(*LSJ*)[17] as a CITE Collection, a collection of lexical-entities. Each object in this collection has three properties:

1. `urn:cite2:hmt:lsj.chicago_md.seq`: The sequence of an entry, because this is an ordered collection.
2. `urn:cite2:hmt:lsj.chicago_md.key`: The headword, or *lemma*, of the lexicon entry.
3. `urn:cite2:hmt:lsj.chicago_md.entry`: The entry itself.

Other projects have encoded the *LSJ* with elaborate markup in TEI-XML, but this collection aims simply to present the lexicon's entries to human readers in a clear and attractive manner. So the data in the `urn:cite2:hmt:lsj.chicago_md.entry:` property, defined as `StringType`, includes Markdown formatting.[18]

For the object identified as `urn:cite2:hmt:lsj.chicago_md:n2389`, the entry property (`urn:cite2:hmt:lsj.chicago_md.entry:n2389`) has this value:

```
**αἴλουρος**, Arist. *HA* 540a10, *Phgn.* 811b9, or αἰέλουρος, ὁ, ἡ, Hdt. and
Comici ll. cc., S. *Ichn.* 296:— `A` **cat, Felis domesticus**, Hdt. 2.66,
Ar. *Ach.* 879, Anaxandr. 39.12, Timocl. 1, LXX *Ep.Je.* 22, Plu. 2.144c.
`A.II` = ἀναγαλλὶς ἡ κυανῆ, Ps.- Dsc. 2.178; also αἰλούρου ὀφθαλμός, ὁ, ibid.
```

But the CITE publication of this data includes a *discoverable data model* identified as `urn:cite2:fufolio:extended_text_properties.v1:`. In the Collection of extended text properties, the property `urn:cite2:hmt:lsj.chicago_md.entry:n2389` is defined as being of the extended-type: `markdown`.

Any application working with this CITE data can ignore that, and will thus render the entry as above, in plain-text. But an application can discover that this property contains Markdown content, and use that information to render the entry with the Markdown transformed:

αἴλουρος, Arist. *HA* 540a10, *Phgn.* 811b9, or αἰέλουρος, ὁ, ἡ, Hdt. and Comici ll. cc., S. *Ichn.* 296:— A **cat, Felis domesticus**, Hdt. 2.66, Ar. *Ach.* 879, Anaxandr. 39.12, Timocl. 1, LXX *Ep.Je.* 22, Plu. 2.144c. A.II = ἀναγαλλὶς ἡ κυανῆ, Ps.- Dsc. 2.178; also αἰλούρου ὀφθαλμός, ὁ, ibid.

17 (Liddell and Scott 1940). For a discussion of this republication of a digital *LSJ*, see C. Blackwell, "Publishing the Liddell & Scott Lexicon via CITE": https://eumaeus.github.io/2018/10/30/lsj.html (last access 2019.01.31).

18 Markdown is a simple standard for applying basic typesetting (emphasis, links, list-formatting) to plain-text documents. See Ovadia (2014) and Voegler et al. (2014).

Other Extended String Text Property types currently in use include `geoJson`, and `teiXml`, but any project is free to identify others. This allows a CITE dataset to include an open-ended number of domain-specific encodings to serve specific needs, but which will all degrade gracefully to plain-text for applications, processes, or readers unaware of those extensions.

The CITE Exchange Format (CEX): plain text serialization of diverse scholarly data

CITE makes no requirements for how these objects, relations, and extensions are captured and stored. Since its origins, CITE data has been stored and served by relational database systems, the Google BigTable database, TEI-XML, RDF in `.ttl` format.[19]

In 2016, Christopher Blackwell, Thomas Köntges, and Neel Smith defined the CITE Exchange Format (CEX), a plain-text, line-oriented data format for serializing citable content following the models of the CITE Architecture. What follows here is a brief overview; the full specification is at https://cite-architecture.github.io/citedx/CEX-spec-3.0.1/.

In a CEX file, distinct types of content are grouped in separate labelled blocks, so that a single CEX source can integrate any content citable in the CITE Architecture.

Blocks are optional (although some blocks may require the presence of one or more other blocks). Authors may limit a CEX serialization to include only those kinds of citable content they choose. A null string or empty text file is a syntactically valid, although empty, CEX data serialization.

1. **Blocks** in a CEX data source are introduced by a line beginning with one of nine **block labels** listed below.
2. Blocks terminate when a new block is introduced or the end of the data source is reached.
3. Content preceding the first labelled block is ignored.
4. Blocks may occur in any sequence in a single CEX serialization.

Valid block labels are:

- `#!cexversion`
- `#!citelibrary`

19 (Chang et al. 2008). "RDF 1.1 Turtle": https://www.w3.org/TR/turtle/ (last access 2019.01.31).

- `#!ctsdata`
- `#!ctscatalog`
- `#!citecollections`
- `#!citeproperties`
- `#!citedata`
- `#!imagedata`
- `#!relations`
- `#!datamodels`

Within a block, the block label is followed by an ordered sequence of lines. That is, while the appearance of blocks in a CEX source is not ordered, line are ordered within each block.

Empty (zero-length) lines are allowed but are ignored. Lines beginning with the string `//` are comments and are ignored. Other lines are treated as the **block contents**.

The syntax of block contents is specific to the type of the block.

CEX affords the ability to share a potentially complex digital library as a single file, independent of any implementing technology. It also allows an expression of an integrated digital library to contain portions of datasets. A teaching edition of a Greek poem might include the poem (as a CTS text), some commentary (as a CITE Collection), and lexical information for the language of the poem. A CEX file could include only those entries from the *LSJ* lexicon that are relevant for the poem, rather than the whole dictionary. By virtue of the CITE2 URNs, those entries would not be separated from their context in the whole lexicon.

A set of demonstration CEX files is published at https://github.com/cite-architecture/citedx.

Code libraries

As of 2018, the definitive implementation of the CITE Architecture is in the code libraries published in the Cite Architecture organization on GitHub.[20] Each of these is written in the Scala[21] language, which allows them to be compiled to `.jar` files for use in the Java virtual machine, or to `.js` files for use in JavaScript/ECMAScript environments.

20 https://github.com/cite-architecture (last access 2019.01.31).
21 https://www.scala-lang.org (last access 2019.01.31).

Each of these libraries depends on SBT, the Scala Build Tool,[22] which allows the library to be compiled and tested, and to have its API documentation generated. That API documentation serves as a definitive definition of the service.

Each library's `README.md` file on GitHub provides instructions for including the library in another project.

Each of these libraries includes tests, which can be run using the Scala Build Tool. These tests constitute a body of documentation complementary to the `scaladoc` API documentation that can (also) be generated using SBT.

The current published libraries are:

Tier 1 Libraries: identification and retrieval

- `xcite`: CTS and CITE2 URN validation and manipulation
- `ohco2`: CTS Texts and corpora thereof
- `citeobj`: CITE Objects and Collections

Tier 2 Libraries: composition

- `cex`: Serializing CITE data to plain-text; generating CITE objects from plain-text serializations.
- `scm`: Scala CITE Manager
- `citerelations`: Subject-Verb-Object relations expressed with 3 URNs.
- `dse`: Documented Scholarly Editions
- `citebinaryimage`: Resolving CITE URNs to images and regions-of-interest on images
- `citejson`: De-marshaling JSON expressions of CITE data into memory representations

Services and applications

- `scs-akka`: A microservice accepting requests via HTTP and returning CITE data marshalled as JSON strings. A page of working examples, drawing on *HMT* data is at http://beta.hpcc.uh.edu/hmt/hmt-microservice/.

22 https://github.com/cite-architecture (last access 2019.01.31).

- `CITE-App`: A ScalaJS web-application that reads data from a CEX file and affords interaction with CITE texts, collections, images, and relations. Because all data is processed in-memory in the browser, this application is suitable only for relatively small and focused libraries. See a working example at http://folio.furman.edu/cite.html.
- `Server-CITE-App`: A version of CITE-App that draws its data from the Akka microservice, and is thus able to work with much larger datasets. The *HMT*'s data is exposed with this application at http://www.homermultitext.org/hmt-digital/.
- `facsimile`: http://www.homermultitext.org/facsimile/index.html. A lightweight application that uses CEX to access a static representation of the *HMT* data. The static representation is a series of Markdown files generated from a CEX library that show citable passages of Iliadic and commentary texts as transcriptions and as ROIs on images of manuscript folios.
- `LSJ Lexicon`: A bespoke application providing access to a CITE representation of *A Greek-English Lexicon*, Henry George Liddell, Robert Scott, revised and augmented throughout by Sir Henry Stuart Jones with the assistance of Roderick McKenzie (Oxford: Clarendon Press. 1940). The lexicon is captured as a CEX file, and served from an instance of the Akka microservice.

Final thoughts

The CITE Architecture arose from the earliest work on the Homer Multitext. In 2000, Gregory Nagy, Casey Dué, and Mary Ebbott began to discuss what a 21st Century edition of the *Iliad* might look like. Their interest was in preserving the tradition of transmission of the text, on the assumption that the details of that transmission hold clues to understanding the nature of Greek epic poetry as the product of an oral tradition of composition in performance. Those details lie in the variations in the text that we find from one manuscript to another, and in particular in Iliadic language quoted in scholarly commentaries from antiquity, and in other authors from antiquity. The editors of the project call these "multiforms" rather than "variants" to emphasize their conviction that these are not divergences from an original, canonical text, but equally legitimate epic expressions.

The edition they proposed would require documenting and aligning many different versions of Iliadic texts, at a fine level of granularity, and aligning those versions to other texts in prose and poetry, to lexical and morphological

data, to digital images, and working with this material in ways that they knew they did not yet imagine.

In 2003, at a conference at the Center for Hellenic Studies of Harvard University, Neel Smith presented a talk entitled "Toward a 'Text Server'," in which he described some of the necessities for rigorous identification and retrieval of texts in a networked digital environment. That was the origin of Canonical Text Services, which was the first component of the CITE Architecture to reach the point of usability. Since 2003, while Neel Smith and Christopher Blackwell have been the main authors of CITE, many others have provided valuable insights, encouragement, wholesome skepticism, and intelligent criticism. An incomplete list of these scholars would include Leonard Muellner, Thomas Martin, Hugh Cayless, Ryan Baumann, Gabriel Weaver, Bridget Almas, Bruce Robertson, Monica Berti, Matteo Romanello, Francesco Mambrini, and Gregory Crane. We would like to recognize the support and inspiration of our late friend, Professor Ross Scaife of the University of Kentucky.

Bibliography

d'Ansse de Villoison, J.-B.-G. (1788): Homeri Ilias. Venetiis: Typis et sumptibus fratrum Coleti.

Chang, F.; Dean, J.; Ghemawat, S.; Hsieh, W.C.; Wallach, D.A.; Burrows, M.; Chandra, T.; Fikes, A.; Gruber, R.E. (2008): "Bigtable: A Distributed Storage System for Structured Data". ACM Transactions on Computer Systems (TOCS) 26:2, 4.

DeRose, S.; Durand, D.; Mylonas, E.; Renear, A. (1990): "What Is Text, Really?". Journal of Computing in Higher Education 1:2, 3–26.

Liddell, H.G.; Scott, R. (eds.) (1940): A Greek-English Lexicon. Revised and augmented throughout by Sir Henry Stuart Jones with the assistance of Roderick McKenzie. Oxford: Clarendon Press.

Ovadia, S. (2014): "Markdown for Librarians and Academics". Behavioral & Social Sciences Librarian 33:2, 120–124.

Smith, D.N.; Weaver, G. (2009): "Applying Domain Knowledge from Structured Citation Formats to Text and Data Mining: Examples Using the CITE Architecture". In: G. Heyer (ed.): Text Mining Services: Building and Applying Text Mining Based Service Infrastructures in Research and Industry. Leipziger Beiträge zur Informatik, Band XIV. Leipzig (reprinted in Dartmouth College Computer Science Technical Report series, TR2009–649, June 2009), 129–139.

Tippett, K.; Rees, M. (2013): "Martin Rees – Cosmic Origami and What We Don't Know". On Being. November 21, 2013. https://onbeing.org/programs/martin-rees-cosmic-origami-and-what-we-dont-know/ (last access 2019.01.31).

Voegler, J.; Bornschein, J.; Weber, G. (2014): "Markdown – A Simple Syntax for Transcription of Accessible Study Materials". In: K. Miesenberger; D. Fels; D. Archambault; P. Peňáz; W. Zagler (eds.): Computers Helping People with Special Needs. ICCHP 2014. Lecture Notes in Computer Science. Volume 8547. Cham: Springer, 545–548.