International Journal on Digital Libraries manuscript No.
(will be inserted by the editor)

A Sharing-Oriented Design Strategy for Networked Knowledge Organization Systems

Ryan Shaw · Adam Rabinowitz · Patrick Golden · Eric Kansa

Received: date / Accepted: date

Abstract Designers of networked knowledge organization systems often follow a service-oriented design strategy, assuming an organizational model where one party outsources clearly delineated business processes to another party. But the logic of outsourcing is a poor fit for some knowledge organization practices. When knowledge organization is understood as a process of exchange among peers, a *sharing*-oriented design strategy makes more sense. As an example of a sharing-oriented strategy for designing NKOS, we describe the design of the PeriodO period gazetteer. We analyze the PeriodO data model, its representation using JSON-LD, and the management of changes to the PeriodO dataset. We conclude by discussing why a sharing-oriented design strategy is appropriate for organizing scholarly knowledge.

Keywords NKOS · periodization · service-oriented architecture · Semantic Web · JSON-LD

This work was generously funded by a Digital Humanities Start-Up Grant from the National Endowment for the Humanities (grant number HD-51864-14).

R. Shaw
School of Information and Library Science
The University of North Carolina at Chapel Hill
E-mail: ryanshaw@unc.edu

A. Rabinowitz
Department of Classics
The University of Texas at Austin
E-mail: arabinow@utexas.edu

P. Golden
School of Information and Library Science
The University of North Carolina at Chapel Hill
E-mail: ptgolden@unc.edu

E. Kansa
Open Context
University of California, Berkeley
E-mail: ekansa@berkeley.edu

1 Introduction

Networked knowledge organization systems (NKOS) make information about concepts, terms, and their relationships accessible over a network. If a NKOS allows programmatic access, tools for authoring, publishing, curating, or finding resources can use its concepts and terms to improve description and querying. NKOS have typically been designed and implemented as Web services. A Web service runs on a remote Web server and responds to requests made by client software running on the user's computer. The tendency to view NKOS as Web services evinces the dominance of a design strategy known as *service-orientation*. A system designed according to this strategy is said to have a *service-oriented architecture* (SOA).

From a technical perspective, SOA refers to a subset of variations upon the typical client-server architectural style for designing Web applications. These include services that expose procedures to be called remotely, services for remote data access via a standard query language, and services that expose representations of resources to be manipulated via HTTP. However, SOA is best understood not as a set of a technical choices, but as the outcome of a broader design strategy. Understood as a strategy, service-orientation is the application of the logic of business outsourcing to software design. By embracing SOA, NKOS architects have (intentionally or not) adopted a view of knowledge organization as a process that can be outsourced to specialist providers.

But the logic of outsourcing is a poor fit for some knowledge organization practices. When knowledge organization is understood as something that can be detached from other "core" activities, a service-oriented strategy for designing NKOS may be appropriate. But when knowledge organization is understood as a process of exchange among peers, a *sharing*-oriented strategy is a better fit. As an example of implementing a sharing-oriented design strategy for NKOS, we describe the design of the PeriodO period gazetteer. We analyze the PeriodO data model, its representation using JSON-LD, and the management of changes to the PeriodO dataset. We conclude by explaining how PeriodO's sharing-oriented architecture benefits the data curators and scholars for whom it was designed.

2 Networked knowledge organization systems

A *knowledge organization system* (KOS) is a tool that can inform its users about concepts of interest in some domain, various names or terms associated with those concepts, and relationships among concepts [19]. Examples include dictionaries, gazetteers, taxonomies, and thesauri. A *networked knowledge organization system* (NKOS) is a KOS accessible over a network, which can be used to facilitate the description, discovery, or retrieval of other networked resources [18, 36]. The term *NKOS* originated with a series of workshops, beginning in 1997, focused on developing a functional and data model for networked knowledge organization systems [38]. Work carried out under the *NKOS* label overlaps with other research programs with similar goals, such as the ISO Topic Maps standards [16] and the broader effort to create a Semantic Web. The basic ideas can be traced back to earlier work on integrating knowledge structures into hypermedia systems [for example 7, 4]. In this paper we use the term *NKOS* to refer broadly to any effort to make information about concepts, terms, and their relationships accessible over a network.

During the past two decades researchers and practitioners have worked through many of the issues involved in constructing NKOS. As a result NKOS have moved from theory to practice. The OCLC library cooperative provides to its members networked access to several vocabularies for describing the form and content of library materials. The Getty Research Institute provides networked access to their vocabularies for describing works of art and architecture. Networked gazetteers such as GeoNames have become critical pieces of infrastructure for translating between place names and geographical coordinates. The successful deployment of these systems reflects a great deal of work on data modeling, semantics, representation, and standards, much of which is documented in the NKOS literature. Yet there has been comparatively little attention paid to the pragmatics of building applications that rely on NKOS data. Why? One reason is a set of assumptions about how NKOS would be integrated into software systems. These assumptions took hold early on and continue to govern the design of NKOS today.

The *S* in *NKOS* is indicative of the assumptions made regarding the integration of NKOS into software systems. The *S* in *KOS* stands for *system* or *structure*. *System* and *structure* name closely related concepts: a system is a set of parts constituting a complex whole, while the arrangement of and relations between these parts constitute the system's structure. A KOS is a system because it consists of parts (concepts and terms) connected together in a consistent and coherent way to form a complex whole. This systematic arrangement of parts defines a certain structure, hence the *S* in *KOS* can just as easily stand for *structure*. The literature on NKOS introduces a third meaning for the *S*: *service*. *Service* is not conceptually related to *system* or *structure*. The introduction of the term *service* thus reflects assumptions about how knowledge systems and their structures should be made available to other applications on the network.

Since the late 1990s researchers have experimented with ways of providing programmatic access to KOS over a network. Binding and Tudhope [3] provide a good overview of this early work, drawing a distinction between systems with an HTML interface via which human users could use a KOS, and Web services allowing programmatic use of a KOS. They note that the conceptualization of NKOS as Web services took hold early on. One of the sessions of the second NKOS workshop, held in 1998, was devoted to sketching out a "functional model of the process of using a Knowledge Organization System (KOS) over a network" [8]. The purpose of this model was to specify, not how KOS content should be displayed or interacted with, but "the interaction over the network between the software on the users' computer (the client) and the software on the computer that manages the reference tool [KOS]" [8]. It was noted during the session, however, that "it is not always easy to draw the line between whether a feature belongs properly to the user interface or the client-server interaction" [8]. Despite this difficulty, the client-server model sketched out here would come to dominate NKOS designs as researchers and IT architects embraced the gospel of SOA.

3 Service-oriented architecture

The term *service-oriented architecture* appears to have been introduced in two technical reports published by the Gartner Group in 1996 [30, 29]. According to Draheim [10], in these reports SOA was characterized primarily as the separation of data-processing logic from data sources. And indeed this is roughly the promise of SOA for NKOS presented by Binding and Tudhope [3]: "a clearer separation of interface components from the underlying data sources, via the use of appropriate Web services." But like so many IT terms, the term *SOA* has a great deal of interpretive flexibility, and definitions have mutated and proliferated over time [10]. In its weakest sense, *SOA* is used to describe any modular software system where the components communicate over a network. A strong definition of SOA, on the other hand, might further specify what it means to conceptualize a component as a *service*. For example, The Open Group [35] defines a *service* as “a logical representation of a repeatable business activity that has a specified outcome.”

Most systems cited as examples of SOA exhibit some derivation of the *client-server* architectural style. The client-server architectural style divides software components into *clients* that act by sending requests and *servers* that respond to those requests [13, 45–46]. This division creates a hierarchy among components, in which a single server will typically respond to requests from multiple clients. Clients and servers can be arranged in multiple layers to achieve “loose coupling” among components, improving their reusability and extensibility[13, 46–47]. In the Gartner reports, *SOA* was used broadly to name this type of layered client-server architectural style. But *SOA* can also imply specific mechanisms via which clients and servers communicate. Early SOA implementations tended to use *remote procedure call* (RPC) mechanisms. These connection mechanisms rely on the specification of a high-level, domain-specific protocol that identifies the procedures that can be called, the parameters they take, and the data they return. For example, the thesaurus service protocols examined by Binding and Tudhope [3] identify specific procedures for querying thesaurus structure such as `get-broader` and `get-narrower` (terms).

In recent years, SOA has come to be more closely identified with *representational state transfer* (REST), an architectural style described and named by by Fielding [13]. RPC-style systems have fallen out of favor as software architects have come to appreciate the benefits of understanding the Web as a system designed in the REST style. REST is a derivation of the client-server style. Where RPC protocols identify procedures, a system in the REST style identifies *resources*, which are abstract conceptualizations of the entities of interest in some domain [13, 88]. The system provides a generic, uniform interface for interacting with and manipulating resources through *representations*. A representation is a document that encodes information about the current or desired state of a resource. A “RESTful” NKOS on the Web, then, is one that *a*) uses URIs to identify terms and concepts, *b*) represents the state of terms and concepts using a standardized document structure, and *c*) enables access to and manipulation of terms and concepts via standard HTTP methods (`GET`, `PUT`, `PATCH`, `POST`, and `DELETE`) .

The SOA paradigm has also come to encompass *linked data* services. Like systems with a REST-style Web architecture, linked data services use URIs to identify entities, and the states of those entities are represented using a standard document structure (typically a serialization of RDF). But many linked data deployments forego the uniform interface of HTTP in favor of a remote data access (RDA) style [26]. RDA-style systems define a standard query language that governs the semantics of client-server communications [13, 49–50]. Instead of multiple access points identified as callable procedures (as in the RPC style), or multiple access points identified as resources (as in the REST style), there is a single access point to which queries are submitted. An example of an RDA-style NKOS is Getty Vocabularies, which provides a SPARQL endpoint to query across the vocabularies for describing art and architecture maintained by the Getty Research Institute. SPARQL endpoints exemplify the RDA style: clients issue SPARQL queries to a remote server, which processes the query to filter, transform, or construct a data set to be returned to the client.

4 Service-oriented design strategy

If RPC-style, REST-style, and RDA-style architectures can all be called *SOA*, what does the term really mean? SOA is best understood not as a specific software architecture, but as the result of following a service-oriented design strategy. Service-oriented design strategy consists of *a*) a set of ideas about what are the desirable properties of software systems, and *b*) a way of thinking about how software systems should provide their functionality . The desirable properties promised by SOA—interoperability, integration, reliability, security, scalability, extensibility, and manageability [2]—clearly reflect the priorities of the “enterprise” software culture primarily responsible for evangelizing it. But the way of thinking about how software systems should provide their functionality also reflects common enterprise practice, specifically the practice of *outsourcing*. Outsourcing involves identifying functions or processes that can be carried out by a specialist provider. If the provider carries out these processes for many different clients, it can achieve economies of scale beyond what any one client could achieve on its own. Furthermore, if the definition of the functions to be provided can be adequately standardized, then their provision can be commodified: clients can easily switch between providers and seek the highest level of service for the lowest cost. Service-oriented design strategy, then, is the logic of outsourcing applied to software: greater efficiency and cost-effectiveness through specialization and standardization.

Evangelists of service-orientation for scholarly and scientific computing explicitly employ the logic of outsourcing: “The last two capabilities—functions and resources—can, in principle, be handed off to specialist providers. If such specialists can deliver resources or operate required functions for many communities, then (again, in principle) economies of scale can be achieved, while scientists can focus on what they are good at—providing content and advancing science” [14, 815]. It is this logic, rather than a specific technology, that NKOS architects have adopted by embracing SOA. Binding and Tudhope [3] make the case succinctly when they suggest that “a provider might offer a KOS service on their portal while independently a DL [digital library] might offer various collections to be searched, offering a choice of vocabulary search tools [from different KOS service providers] or allowing the user to choose if a standard protocol existed." These are the fundamental ideas that have guided NKOS development over the past two decades: that knowledge organization is a service that can be offered by a specialist provider, that efficiencies and economies of scale can be achieved by centralizing the provision of knowledge organization, and that knowledge organization can be commodified through standardized protocols, representations, or query languages. Thus despite differences over time in the specifics of the preferred style of implementation (RPC, REST, or RDA), service-orientation as a strategy for organizing the interactions of KOS users and providers has remained stable.

A service-oriented strategy makes sense when designing NKOS intended to establish consensus around a system of concepts and terminology, with the goal of increased efficiency. For example, large enterprises that buy from and sell to one another may standardize the definitions and names of the categories of products bought and sold. In such cases it is sensible to conceive of the provision and maintenance of the standard taxonomy as a business process that can be outsourced to specialist providers. The specialist providers, supported perhaps by fees from enterprises wishing to participate in the standardization process, can take advantage of economies of scale to manage the shared taxonomy more cheaply than the enterprises could do individually. The enterprises can then focus on providing business value and making profits, rather than taxonomy management.

5 Sharing-oriented design strategy and architecture

But not all KOS are intended to standardize concepts and terminology. Some KOS may be intended to map a diverse conceptual and terminological landscape rather than provide canonical names and definitions. In section 6 we describe one such KOS, which records definitions of historical periods authored by archaeologists and other scholars. These scholars' definitions reflect their individual data and interpretations; they are not necessarily seeking to establish consensus. Accordingly the logic of outsourcing does not apply: these scholars should not "hand over" management of their definitions; they themselves are the specialists best positioned to manage them. The judgments they express through definition are their own and cannot be made for them by an external "service provider." However, these scholars do have an interest in disseminating information about changes to their definitions and keeping up-to-date with the definitions used by others. What they need is a NKOS that promotes dissemination of concepts and terminology, without assuming that such dissemination requires entering into an ongoing outsourcing relationship with external service providers. An alternative design strategy is needed: a sharing-oriented design strategy.

A sharing-oriented design strategy does not differ significantly from a service-oriented strategy in terms of objectives: interoperability, integration, reliability, security, scalability, extensibility, and manageability are still desirable. The significant change is in the way of thinking about how these objectives should be reached. A sharing-oriented design strategy for scholarly NKOS does not attempt to isolate the management of terms and concepts as a specialized activity, but instead treats it as intimately interconnected with the broader process of scholarship. Scalability is achieved not by enabling a single "service provider" to work for many clients, but by streamlining the means of communication among peers. Consensus on a standardized "single source of truth" is explicitly rejected as a goal, in favor of better tools for managing diverse perspectives on reality. Likewise some efficiency is sacrificed for greater flexibility and autonomy.

A sharing-oriented design strategy does not necessarily dictate a single specific architecture for NKOS. But one architecture that has proven to be compatible with a sharing-oriented strategy is that of distributed version control systems. Distributed version control allows people to collaborate in a purely peer-to-peer, distributed fashion through the exchange of *patches*: documents describing local changes to data. Distributed collaboration via the sharing of patches has proved quite successful among software developers, most recently in the form of the free distributed version control system Git [6] and the commercial repository hosting service GitHub. Taking distributed version control as a model leads to a sharing-oriented architecture for NKOS that enables the following workflow:

1. Obtain a copy of a KOS serialized in a standard format, or author a new KOS from scratch.
2. Work with one's KOS, making changes as needed to suit one's purposes.
3. When one wishes to give changes or updates to one's KOS to someone else:
   (a) Obtain a copy of that person's KOS.
   (b) Generate a patch describing the differences between one's own KOS and that person's KOS.
   (c) Send the patch to that person, along with whatever explanation or argumentation ones wishes to add. That person can then choose to apply or reject the patch. If they accept it, there may be conflicts that will need to be resolved before the accepted patch can be applied.

Note that nothing about this workflow requires the notion of a canonical or authoritative version of a KOS. However it is often useful to be able to reference a canonical version of a KOS, preferably one that makes some guarantees about the quality of its data over time. In the section 6 we describe in detail the sharing-oriented architecture of the PeriodO period gazetteer, including the use of persistent identifiers for referencing a canonical version of the PeriodO dataset.

6 The PeriodO period gazetteer

The PeriodO period gazetteer[1] documents definitions of historical period names. Each entry of the gazetteer identifies the definition of a single period. To be included in the gazetteer, a definition must *a*) give the period a name, *b*) impose some temporal bounds on the period, *c*) have some implicit or explicit association with a geographical region, and *d*) have been formally or informally published in some citable source. Period definitions from the same citable text or dataset are grouped into collections. Much care has been put into giving period definitions stable identifiers, associating our documentation of the definition and its source with a string usable in citations. Anyone can propose additions of new definitions to PeriodO, and anyone can use PeriodO identifiers to refer to a specific definition of a period in PeriodO.

Currently PeriodO is focused on periods defined by archaeologists. PeriodO has documentation of 1,800 period definitions at the time of writing, all of which appeared in sources published or cited by archaeologists. The 1,800 definitions were gathered from around 70 sources. About 1,000 of the definitions come from projects curating and publishing archaeological data on the Web, such as the Fasti Online database of archaeological excavations[2] and the Levantine Ceramics Project[3]. These curation projects provide interfaces for browsing and comparing data from different sources. Comparison along the temporal dimension is critical for such interfaces, and so there is a need to relate the period names used to label data to temporal ranges. As a result these projects have created definitions of the period names they use, definitions that include descriptions of the temporal ranges encompassed by the periods named. The source contributing the greatest number of such definitions (around 500) is the Digital Index of North American Archaeology (DINAA), an effort to build a union index of databases of archaeological sites in North America.

Most of the remaining definitions were gathered from archaeological monographs, articles, and textbooks by the GeoDia project [27]. GeoDia was an attempt to address the gap between the heterogeneous and spatially-situated scholarly usage of period terms and their homogeneous and standardized appearance in textbooks and reference materials. Art-history textbooks, for example, usually present a single temporal definition for the term *Archaic* with reference to ancient Greek culture [e.g. 34]; but a review of the art-historical and archaeological literature reveals that the same term is used by different authoritative sources to refer to different date-ranges in different geographic locations (and sometimes even in the same locations). Rather than creating a standardized periodization, then, the GeoDia project set out to document and visualize the usage of period terms on a site-by-site basis. The project collected a broad range of definitions of period terms from authoritative archaeological texts that discussed particular sites or regions, with a focus on the ancient Mediterranean. The GeoDia project and its influence on the design of PeriodO are discussed in more detail in section 6.1.

The PeriodO dataset is already a useful resource, as it enables comparison of and mapping between periods defined by several major archaeological databases, and documents hundreds more such definitions in archaeological literature. We hope that the PeriodO data will prove useful as a tool for the data curators who contributed period definitions. And we hope that PeriodO will be adopted by scholars and researchers as a way to publicly document their judgments about historical periods. Thinking about how to provide both a practical tool for data curators and a convenient system for public documentation led us to develop the sharing-oriented design strategy outlined in the section 5.

6.1 Period definitions and collections

Understanding the design of the PeriodO data model requires understanding the process through which the PeriodO dataset was created. As discussed above, much of the PeriodO dataset was initially created by Adam Rabinowitz and his assistants for the GeoDia spatial timeline of the ancient Mediterranean. A goal of GeoDia was to help students become "aware of the multiplicity of terms and concepts that are used to group material remains into chronological periods or geographic units" [27]. Toward that end, the GeoDia project collected definitions of period terms from authoritative archaeological texts, focusing on the ancient Mediterranean. For example, Lorrio and Zapatero [24]'s article "The Celts in Iberia: An Overview", published in 2005 in the online journal *e-Keltoi*, proposes a certain perspective on the periodization of the Iberian peninsula in antiquity. As is so often the case in the writing of history, the expression of this perspective includes the explicit temporal and geographic definition of period terms. In this case, Lorrio and Zapatero use the terms *Late Bronze Age II/III*, *Early Iron Age*, and *Late Iron Age*. Only the latter two are given clear definitions in the text of the article:

[1] The canonical dataset is `http://n2t.net/ark:/99152/p0`. Public domain source code is at `https://github.com/periodo`.

[2] `http://www.fastionline.org/`

[3] `http://www.levantineceramics.org/`

- "Previously, this region [the central Douro Valley] had been inhabited by the Soto de Medinilla group (ca. 800-400 BC), which defined the Early Iron Age" (217)
- "Between 800 and 400 BC (the Early Iron Age), the Castro Culture communities further developed the tendencies of the previous period [the Late Bronze Age] ..." (222)
- "The Late Iron Age (400-100 BC) witnessed the emergence of more unequal and complex societies, the regional compartmentalization of land and a strong differentiation of material culture ..." (223)

These definitions merited inclusion in GeoDia because they include descriptions of both temporal coverage (expressed explicitly via phrases such as "between 800 and 400 BC") and spatial coverage (implicitly via the focus of the article on the Iberian Peninsula).

The GeoDia project thus established the initial model of grounding period definition in scholarly assertions, a model that has been refined in PeriodO. A minimal PeriodO period definition consists of a human-readable label and statements of temporal and spatial coverage. The preferred label and statements of temporal and spatial coverage are taken verbatim from the text of the source, where possible. These textual labels are then supplemented with normalized values. Descriptions of temporal coverage are parsed into standardized forms (ISO 8601 lexical representations of Gregorian calendar years). Descriptions of spatial coverage are supplemented by one or more references to *spatial things* (things having spatial extent), typically modern nation-states, but occasionally more specific spatial entities. For example, the spatial coverage of Lorrio and Zapatero's definition of the *Early Iron Age* is described textually by the label `Galicia`, and this label is supplemented by the URL of the DBpedia resource `http://dbpedia.org/resource/Galicia_(Spain)`. Where periods are originally defined in languages other than English, English-language labels are assigned as alternate labels.

Definitions from the same source are grouped into collections. Thus Lorrio and Zapatero's definitions are grouped into a period collection that is associated with bibliographic metadata describing the source article. Where possible, the source bibliographic entity for a period collection is identified with a URL, typically a WorldCat URL for a book or a CrossRef DOI for a journal article. If no stable URL providing RDF metadata has been assigned to the source bibliographic entity, we record in PeriodO the title, names of creators, year published, and either a formatted citation or a (non-metadata-providing) URL for the source. A period collection is not a periodization, where *periodization* is understood as meaning a single coherent, continuous division of historical time, each part of which is labeled with a period term. A set of period term definitions *may* constitute a periodization, but this is not necessary. For example, a monograph could propose and compare two alternative periodizations of the history of some region. Period definitions extracted from the monograph would share the same source and would thus be grouped together in a PeriodO collection. However, this grouping would not reflect the fact that the periods defined belong to two distinct periodizations. Further assertions making the distinction could be added to the PeriodO dataset at a later point. Our initial goal has been to identify period definitions having a common source, and to leave more complex modeling of periodizations to future work, if it is deemed necessary.

We have formally modeled PeriodO period definitions as SKOS concepts, and the period collections (into which definitions are grouped) are modeled as SKOS concept schemes [25]. The PeriodO dataset itself is an unordered container (`rdf:Bag`) of concept schemes. The `dcterms:source` of each concept scheme is the bibliographic entity from which the period concepts were extracted. Each period definition, in addition to being modeled as a SKOS concept, is also modeled as an OWL-TIME *proper interval*, which is an interval of time with distinct beginning and end points [20]. OWL-TIME defines predicates corresponding to the possible binary relations between temporal intervals identified by Allen and Ferguson [1]. We use the OWL-TIME predicates `intervalStartedBy` and `intervalFinishedBy` to link period definitions to the proper intervals that represent their beginnings and ends. Period beginnings and ends are themselves represented as proper intervals; they are not instants but always have some temporal extent reflecting the intended degree of precision. Each proper interval representing a period beginning or end has a label taken from the original source (e.g. `800 B.C.`), and a `DatetimeDescription` providing an interpretation of the interval in terms of individual properties. Typically this datetime description has only a single property, `time:year`, the value of which is an `xsd:gYear` value (an ISO 8601 lexical representation of a Gregorian calendar year). For example, an interval with the label `800 B.C.` would have a `time:year` property with the value `-0799`.

In some cases, `time:year` is insufficient for describing the interval. For example, Fouilland et al [15] define the *Fase II* period for the necropolis of Monte Casasia in Sicily as starting at the "beginning of the 6th century B.C.", so this phrase is used as the label of the interval beginning this period. If the start of the period had been defined as "600 B.C.", then a datetime description with a `time:year` value of `-0599` may have been sufficient. But "beginning of the 6th century B.C." seems less precise than "600 B.C.", and so to reflect the lack of precision in this descriptive phrase, we indicate the temporal extent of the interval using two properties, `earliestYear` and `latestYear`, which are assigned the values `-0599` and `-0566` respectively. The 33-year extent indicated by this datetime description, unlike the interval's label, should not be ascribed to the creators of the original source. It is a description assigned by the PeriodO maintainers to the interval for use when ordering, querying, or visualizing period definitions by their temporal coverage.

6.2 JSON-LD representation

One of our design goals for PeriodO was that the dataset be easy to work with programmatically. We want to encourage others to build tools that consume PeriodO data. We decided that this required us to publish PeriodO data as JavaScript Object Notation (JSON). JSON is a subset of the JavaScript programming language intended for use as a data exchange format. Data is represented as combinations of JavaScript *objects* (sets of key-value pairs) and *arrays* (ordered lists of values). Values may themselves be objects or arrays. Every programming language has libraries for reading and encoding JSON, and such libraries are included in the core distributions of most major languages. And of course JSON is the format of choice for working with data in browser-based applications, using libraries such as Backbone or D3.js. However, many of our partners used RDF-based tool stacks for working with data, and making the dataset easy to work with programmatically for them meant publishing PeriodO data as Linked Data. Fortunately, the recent W3C Recommendation of JSON-LD, a JSON-based serialization for Linked Data [32], made it possible to design a data format that is easily programmable using either a JSON-based or a RDF-based tool stack.

JSON-LD documents are, syntactically, just JSON documents: any ordinary JSON parser can parse a JSON-LD document. However a JSON-LD document, unlike an ordinary JSON document, follows certain conventions that allow it to be interpreted as encoding an RDF graph. A JSON-LD node object (set of key-value pairs) is interpreted as representing a node in an RDF graph. Each key-value pair in the object represents a predicate and a value or, if the value is an array, an unordered set of predicate-value pairs sharing the same predicate but having different values. If a value is an object, it is interpreted as another node in the graph, otherwise it is interpreted as a literal. Finally, JSON-LD documents use reserved keywords to further guide interpretation of the JSON data structures as RDF. The `@id` keyword, when used as the key in a JSON-LD node object, indicates that the value associated with it should be interpreted as a URI identifying the node in the RDF graph. A `@context` key has as its value a JSON-LD context object, which shows how terms used as keys in node objects should be interpreted as predicates with specific URIs.

Figure 1 shows how a PeriodO period definition is represented using JSON-LD. The JSON-LD representation of the PeriodO data consists of a single root JSON-LD node object. The root node object has a `periodCollections` property, the value of which is a large object representing all the period collections in the PeriodO dataset. The keys of this object are the identifiers for the period collections, while the values are node objects, each representing an individual collection. Each node object representing a period collection has a `source` property, the value of which is an object containing bibliographic metadata describing the source, including (if possible) an external URI for the source (e.g. a WorldCat or CrossRef Linked Data URI). The value of the `definitions` property is another index object, this time with individual period definition identifiers as keys and the node objects representing those definitions as values.

Structuring the dataset as an object like this makes it efficient to use as an index for looking up period collections by their identifiers. However we do not want these index keys to be interpreted by a JSON-LD processor as predicate URIs, so we use the JSON-LD `@container` and `@index` keywords to indicate that the value of the `periodCollections` term is an index. The keys of the index are thus ignored by JSON-LD processors, and the values are interpreted as objects of the predicate mapped to the `periodCollections` terms, which is simply `rdfs:member`. Structuring the dataset as an object, in addition to providing efficient indexing, also simplifies the JSON Patch [5] data structures used to describe changes to the dataset. Since JSON objects are unordered sets of key-value pairs, patches changing them can consist of `add`, `remove`, or `replace` operations and needn't use `move` operations to keep elements in order.

6.3 Managing the canonical version

The canonical version of the PeriodO dataset is *canonical* only because the principals involved in the PeriodO project have made a commitment to maintain long-term HTTP access to PeriodO dataset, and to review submitted patches to the dataset for completeness and correctness. It is entirely possible that there could eventually be multiple independent projects making such commitments, perhaps oriented toward different scholarly sub-disciplines with needs for identifiable period definitions. At the time of writing, however, there is only a single such project, and the dataset maintained by the project is the one we are deeming *canonical*. Establishing a canonical version of the PeriodO dataset requires *a*) a way to identify and document the current version of the dataset and past changes to it, and *b*) a process for accepting, reviewing, and applying or rejecting proposed changes to the canonical dataset .

6.3.1 Persistent identifiers

To identify the canonical dataset and the concepts it describes, and to provide reliable long-term HTTP access to the canonical dataset, we rely on the ARK identifier scheme

```
{ "@context": { JSON-LD context elided },
  "id": "p0d/#periodCollections",
  "type": "rdf:Bag"
  "periodCollections": {
    other period collections elided
    "p0fh3zc": {                                                        period collection
      "id": "p0fh3zc",
      "type": "PeriodCollection"
      "source": {
        "title": "The Celts in Iberia: An Overview",
        "creators": [
          { "name": "Alberto J. Lorrio" },
          { "name": "Gonzalo Ruiz Zapatero" } ],
        "yearPublished": 2005,
        "url": "http://www.uwm.edu/celtic/ekeltoi/volumes/vol6/6_4/lorrio_zapatero_6_4.html",
      },
      "definitions": {
        other period definitions elided
        "p0fh3zcqs6h": {                                  period definition
          "id": "p0fh3zcqs6h",
          "type": "PeriodDefinition",
          "label": "Early Iron Age",
          "localizedLabels": { "eng-latn":["Early Iron Age"]},
          "spatialCoverage": [
            { "id": "http://dbpedia.org/resource/Galicia_(Spain)",
              "label": "Galicia" }
          ],
          "start": {
            "in": { "year": "-0799" },
            "label": "800 B.C."
          },
          "stop": {
            "in": { "year": "-0399" },
            "label": "400 B.C."

}}}}}
```

Fig. 1: A PeriodO period definition represented using JSON-LD.

[22] and the California Digital Library's EZID service [33]. To manage the acceptance, review, and application or rejection of proposed changes, we have built a simple HTTP API. This API can be used directly and programmatically, or via a JavaScript client published alongside the PeriodO dataset. The rest of this section describes PeriodO identifiers and the PeriodO HTTP API in more detail.

In the JSON-LD serialization of the canonical PeriodO dataset, period definitions and collections of period definitions are identified by keys in the JSON-LD object, such as `p0fh3zc` for a period collection or `p0fh3zcqs6h` for a period definition within that collection. These short sequences are guaranteed to uniquely identify records within a single local copy of the PeriodO dataset. To uniquely refer to period definitions or collections in the canonical dataset, the short local key must be appended to the canonical dataset's globally unique prefix. The canonical PeriodO dataset uses the globally unique prefix `ark:/99152/`. The `ark:` label indicates that canonical PeriodO identifiers are Archival Resource Keys or ARKs [23].

PeriodO identifiers are opaque and hence not useful on their own. For an opaque identifier to be used effective, there must be a straightforward way to obtain a description of the identified object [23]. The CDL provides the resolution of identifiers to descriptions as a service, currently located at `http://n2t.net/`. This URL locates a *name mapping authority hostport* (NMAH). The NMAH used may change over time; there can be multiple alternative name mapping authorities. `ark:/99152/p0` is registered as an identifier of the PeriodO Period Gazetteer with the California Digital Library's EZID service for the maintenance of long-term identifiers. EZID maintains a public index of registered identifiers, including descriptions of the things identified. The description of the PeriodO Period Gazetteer includes a URL locating the HTTPS server that is responsible for both serving the PeriodO dataset and receiving patches. Hence EZID responds to requests to (for example) `http://n2t.net/ark:/99152/p0fh3zcqs6h` with an HTTP `302 Found` status, and a `Location` header with the URL where this resource can currently (but not permanently) be found [12, section 6.4.3]. Clients can then make a new request to this temporary URL.

6.3.2 Proposing and reviewing modifications

In section 5 we described the sharing-oriented process for working with PeriodO data: make a local copy that can be modified at will, and exchange proposed modifications via patches (see figure 2). Rather than exposing a granular API or a SPARQL endpoint, we make the entire PeriodO dataset available as an HTTP `PATCH`able resource [11]. A contributor proposes modifications by sending an HTTP `PATCH` request to a PeriodO server, including an `Authorization` header with a valid and unexpired bearer token [21]. Currently the only way for a contributor to obtain a bearer token is to create an ORCID [17] identifying herself and to grant the PeriodO project permission to identify her using her ORCID. After granting this permission the server will respond with a version of the PeriodO client application that has possession of the bearer token. Requests made to the PeriodO API with this bearer token are then assumed to have originated with the contributor who granted permission. Thus if the bearer token is leaked, others will be able to submit patches and falsely attribute them to her. The PeriodO API uses HTTPS to avoid transmitting bearer tokens openly, but given that there are various other ways a bearer token could leak, this should not be considered a secure authentication scheme. In the event of a bearer token leak, the token can be revoked by either PeriodO administrators or the person on whose behalf the token was originally created.

Assuming it has been properly authenticated, the body of the PATCH request must be a valid JSON Patch document [5]. Figure 3 shows a JSON Patch document describing a correction made to a single period definition. If the request body is not a valid JSON Patch document, or if the patch is valid but cannot be applied to the current version of the dataset (perhaps due to intervening modifications), the server will not process the request. Otherwise, the server will respond with a `202 Accepted` status code, indicating that the patch has been accepted for processing but nothing further has happened. The response will include a `Location` header giving the URL for the accepted patch. The current status of the patch is published at this URL. At some point later, one of the PeriodO curators will examine the patch and decide whether or not to apply it to the dataset. If the curator decides to apply the patch, identifiers are minted for any new period definitions or collections introduced by the patch. The dataset is then copied and the patch is applied to the new copy, which becomes the current dataset. All past versions of the dataset, as well as all patches that have been accepted (whether applied, rejected, or yet to be reviewed), are readable through the HTTP API.

The PeriodO client is a JavaScript application intended to run in a Web browser. The client provides an interface for browsing the period definition collections and comparing period definitions, and for modifying and adding new period definitions and collections (see figure 4). It can also be used offline as a tool for browsing and editing PeriodO data from local files. This allows us to archive copies of the PeriodO client alongside snapshots of the dataset, suitable for deposit in a long-term scholarly repository. When a request is made to the PeriodO server for an HTML representation of the PeriodO data, the response includes the code for the PeriodO client. When the client begins running in the browser, it requests a copy of the PeriodO dataset from the same server, and stores this copy locally. At this point the client can be used entirely offline, unless one wishes to propose modifications to the data. If a contributor wishes to propose modifications to the canonical version of the PeriodO dataset, or any other version being hosted by a PeriodO server, she must use the version of the PeriodO client provided by that specific server, so that she can be authenticated and identified as the author of the patch.

7 Discussion

The design of PeriodO reflects our years of experience consuming and publishing data on the Web. First, it is important not to introduce unnecessary external dependencies. Using someone else's data always introduces a dependency on the data provider. But if one has a local copy of the data, one is dependent on the provider only for the initial dataset and periodic updates, rather than for every query over the data. Using the data becomes more like using a software library than a real-time service relationship. And in the event the data providers abandon their stewardship of the data, the users still have the data itself to do with as they wish. So unless the dataset is so large and complex that it requires special resources, such as a cluster of servers for processing queries, it is preferable to download a local copy of the data than to access it via a Web service.

Second, it is difficult to commit to reliably maintaining even a simple information service for the long term. Security patches must be kept up to date. Bugs must be fixed. Domain names and SSL certificates must be renewed. If the information service is computationally intensive, as RDA-style services such as SPARQL endpoints can be, resources must be carefully managed. Queries that would consume too

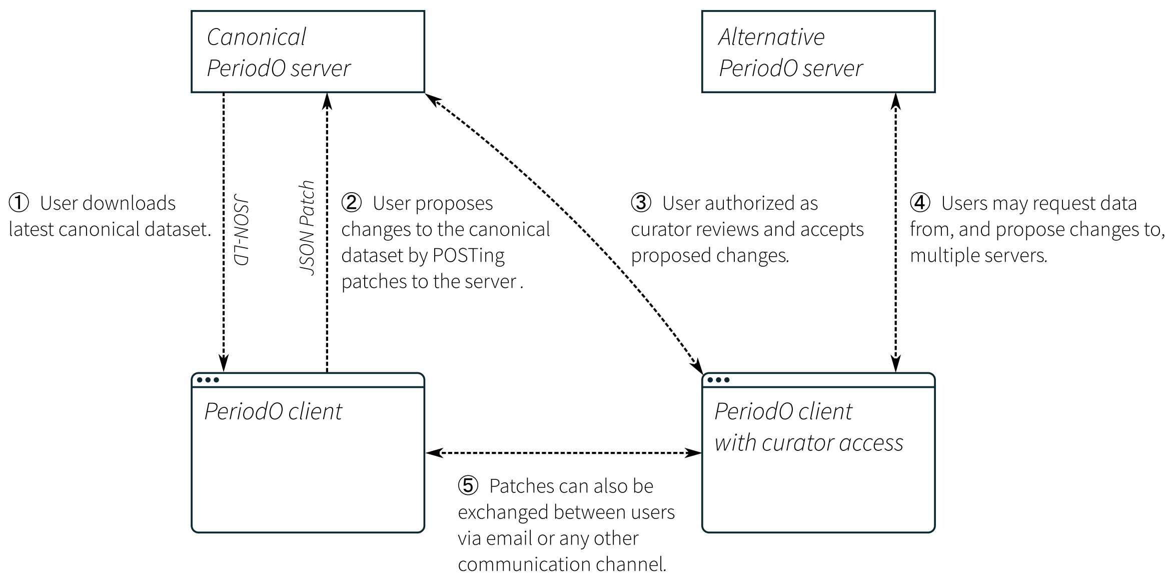

Canonical PeriodO server
Alternative PeriodO server
① User downloads latest canonical dataset.
JSON-LD
JSON Patch
② User proposes changes to the canonical dataset by POSTing patches to the server.
③ User authorized as curator reviews and accepts proposed changes.
④ Users may request data from, and propose changes to, multiple servers.
PeriodO client
PeriodO client with curator access
⑤ Patches can also be exchanged between users via email or any other communication channel.

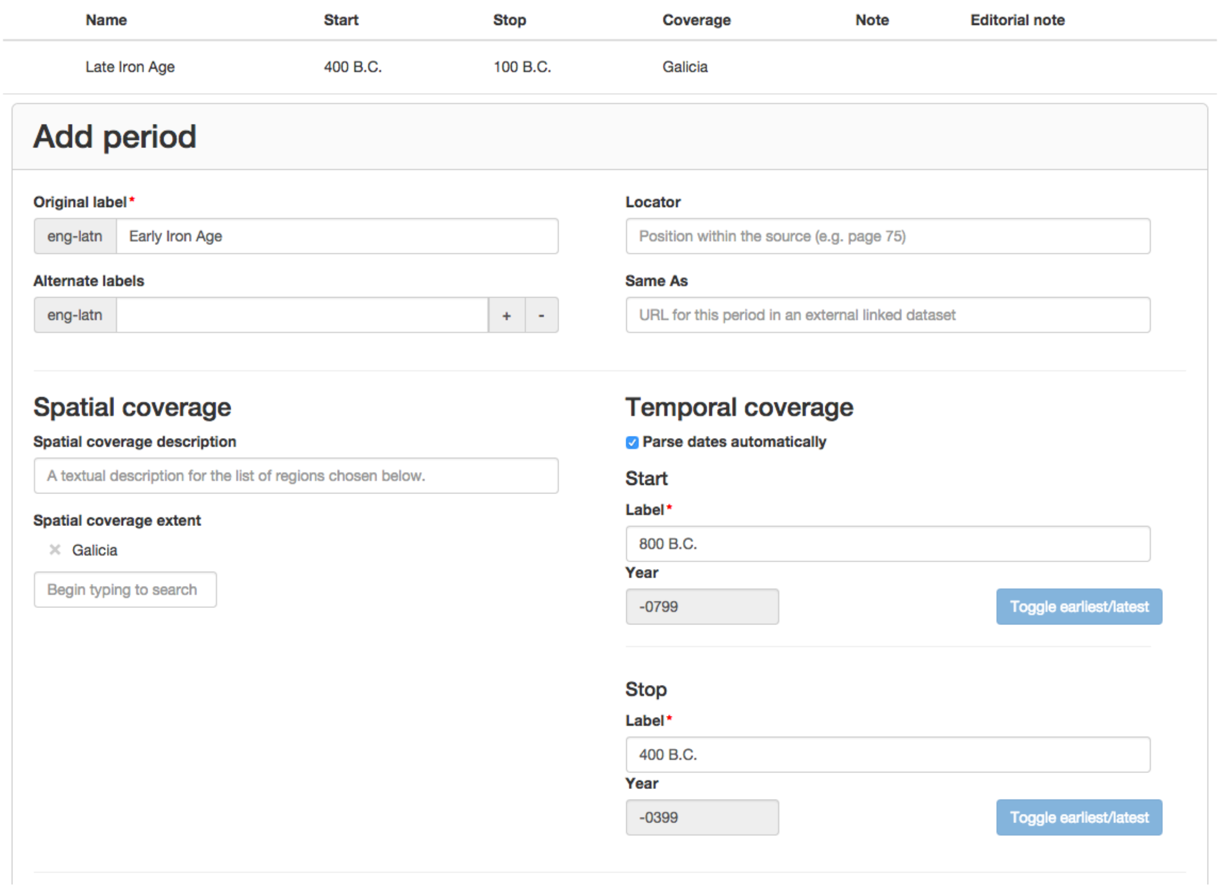
Fig. 2: The sharing-oriented process for working with PeriodO data.

```
[
  { "path": "/periodCollections/p0fh3zc/definitions/p0fh3zcqs6h/stop",
    "op": "add",
    "value": { "in": { "year": "-0449" }, "label": "450 B.C." }
  },
  { "path": "/periodCollections/p0fh3zc/definitions/p0fh3zcqs6h/editorialNote",
    "op": "add",
    "value": "The end of this period was incorrect due to a data entry error."
  }
]
```

Fig. 3: A JSON Patch document describing a correction made to a single period definition.

much memory or CPU must be prevented. And these are simply the more technical challenges of keeping an information service available. More formidable are the organizational challenges: who has SSH access to the server? Who is responsible for making decisions about access and versioning? While designing PeriodO we sought to reduce the commitment of maintaining PeriodO to the bare minimum. Use of PeriodO data and the tools for browsing, visualizing, and editing PeriodO data does not require any PeriodO server to be running.

The PeriodO architecture is a radical response to what Ren and Lyytinen [28, 79] identify as the "fundamental challenge" of deploying information services: "to determine an economic level of granularity for reuse: how fine or coarse should a service be". Binding and Tudhope [3] acknowledge this challenge, noting that if a thesaurus service has too fine-grained of an access protocol—for example, one specifying procedures for getting data about individual concepts and terms—using it will be unacceptable slow, as most interfaces will need to make multiple calls to gather the data needed to generate browsing interfaces. They recommended that thesaurus services provide coarse-grained "composite" procedures allowing necessary data to be obtained with a single request. The coarsest possible response to a query over a dataset is the entire dataset itself. In 2004, expecting clients to cache a local copy of an entire KOS for local querying may have been unrealistic. Now it is feasible to store datasets with hundreds of thousands of records in browser storage, and query them quickly using JavaScript [31]. Projects like DBpedia take advantage of this fact to



Name	Start	Stop	Coverage	Note	Editorial note
Late Iron Age	400 B.C.	100 B.C.	Galicia		

Add period

Original label*: eng-latn Early Iron Age

Locator: Position within the source (e.g. page 75)

Alternate labels: eng-latn + -

Same As: URL for this period in an external linked dataset

Spatial coverage

Spatial coverage description: A textual description for the list of regions chosen below.

Spatial coverage extent: × Galicia

Begin typing to search

Temporal coverage

☑ Parse dates automatically

Start

Label*: 800 B.C.

Year: -0799

Toggle earliest/latest

Stop

Label*: 400 B.C.

Year: -0399

Toggle earliest/latest

Fig. 4: Editing a period definition in the PeriodO client.

increase the availability of its data: its "triple fragment patterns" interface uses an approach similar to PeriodO, pushing the querying process from the server to the client [37].

By eschewing a service-oriented architecture, the design of PeriodO also avoided some of the pitfalls of a service-oriented design process. In a service-oriented design process, businesses practices are analyzed in an attempt to identify common repeated activities. These activities can then be logically represented as services, and the practices to be supported can be re-conceptualized as compositions of those services. The goal is to abstractly represent commonly repeated activities so that they can be outsourced, and their provision eventually commodified. But scholarly practices, especially in the humanities, resist this kind of abstraction and decontextualization, as the organizers of Project Bamboo discovered. The Bamboo planning process was nearly derailed when participating scholars objected to the assumption that scholarship can be reduced to an abstract recipe [9]. The problem is not that scholars are unable to analyze their own practices, but that those practices are not easily detached from the contexts of individual researchers and their fields of study. Note-taking, for example, is an activity observed across a wide range of scholarly practices, but the relationship of note-taking to other scholarly activities varies from scholar to scholar. Note-taking is a repeated activity, but it does not have a specified outcome and it is not easily described in the abstract.

Rather than attempt to abstractly describe scholarly activities in order to impose a division of labor, PeriodO focuses on representing scholarly assertions. Representing assertions has its own challenges, but they are familiar ones, as this is exactly what the designers of KOS have been doing for centuries. A gazetteer, for example, is a curated collection of scholarly assertions about places. Scholars can reach consensus on what pieces of information are needed in a gazetteer entry more easily than they can articulate the possible ways they might incorporate that gazetteer into their scholarly practice. The PeriodO architecture was designed to facilitate the public documentation and peer-to-peer sharing of such scholarly assertions. It is a sharing-oriented architecture in which participants collaborate by exchanging representations of assertions using a shared data format. The design of PeriodO deliberately discourages the establishment of an ongoing service relationship between the PeriodO project and its users. Instead of being a service provider with the commitments that entails, the PeriodO project is just another peer with which period assertions can be shared. It is distinguished from other peers by the fact that it has committed to a trustworthy curation process, and nothing restricts other peers from making similar commitments.

8 Conclusion

When knowledge organization involves the exchange of assertions among peers, a sharing-oriented architecture makes more sense than a service-oriented one. Scholarly communication involves various forms of knowledge organization. Some of these forms of knowledge organization involve a division of labor, such as the division of labor between scholars who write books and the librarians who make them available in libraries. But when scholars share less formally published products such as datasets, software, or working notes, there is no clear division of labor. These kinds of scholarly products are vehicles of peer-to-peer communication among scholars. Differences in the concepts and terminology they employ cannot be standardized away, as it is these very differences that are often at stake in such forms of scholarly communication. Scholars need better ways to exchange information about the different concepts they develop and the different terminology they employ. Sharing-oriented NKOS like PeriodO can help provide them.

PeriodO provides tools for authoring, organizing, and sharing scholarly assertions about period concepts. If these tools are adopted widely, scholarly products will begin to refer to these assertions using PeriodO identifiers. Once we can begin harvesting PeriodO identifiers from scholarly outputs “in the wild,” a new phase in the development of PeriodO will be possible. Alongside the period gazetteer documenting definitions of historical period names, we can build a period-focused index of scholarly work. Where the period gazetteer allows scholars to point to specific period definitions, the period-focused index would point in the opposite direction: from specific definitions of period names to specific resources that use those definitions. Given a sufficiently large number of resources citing PeriodO definitions, it may even be possible to train statistical classifiers to suggest suitable period names for annotating unlabeled resources, or to disambiguate period names in texts by linking them to PeriodO definitions. First, however, we must make it as easy and attractive as possible for data curators and scholars to adopt PeriodO. We expect that PeriodO’s sharing-oriented architecture will play an important role in facilitating this.

Acknowledgements We are grateful to the National Endowment for the Humanities for funding this work, to our partners who contributed period definitions and provided feedback, and to Sarah Buchanan for her work compiling the initial PeriodO dataset.

References

1. Allen JF, Ferguson G (1994) Actions and Events in Interval Temporal Logic. Journal of Logic and Computation 4(5):531–579, DOI 10.1093/logcom/4.5.531, URL `http://logcom.oxfordjournals.org/cgi/doi/10.1093/logcom/4.5.531`
2. Austin D, Barbir A, Ferris C, Garg S (2004) Web Services Architecture Requirements. Tech. rep., W3C, URL `http://www.w3.org/TR/wsa-reqs/`
3. Binding C, Tudhope D (2004) KOS at your Service: Programmatic Access to Knowledge Organisation Systems. Journal of Digital Information 4(4)
4. Bruza PD (1990) Hyperindices: A Novel Aid for Searching in Hypermedia. In: Proceedings of the First European Conference on Hypertext, Cambridge University Press, Versailles, France, pp 109–122
5. Bryan P, Nottingham M (2013) RFC6902: JavaScript Object Notation (JSON) Patch. Tech. rep., IETF, URL `https://tools.ietf.org/html/rfc6902`
6. Chacon S, Straub B (2014) Pro Git, 2nd edn. Apress, URL `http://git-scm.com/book/en/v2`
7. Collier GH (1987) Thoth-II. In: Proceeding of the ACM conference on Hypertext - HYPERTEXT ’87, ACM Press, New York, New York, USA, pp 269–289, DOI 10.1145/317426.317446, URL `http://dx.doi.org/10.1145/317426.317446`
8. Davies R (1998) Working Session 2: Functional Model. URL `http://nkos.slis.kent.edu/SESS2.html`
9. Dombrowski Q (2014) What Ever Happened to Project Bamboo? Literary and Linguistic Computing 29(3):326–339, DOI 10.1093/llc/fqu026, URL `http://llc.oxfordjournals.org/cgi/doi/10.1093/llc/fqu026`
10. Draheim D (2010) The Service-Oriented Metaphor Deciphered. Journal of Computing Science and Engineering 4(4):253–275, DOI 10.5626/JCSE.2010.4.4.253, URL `dx.doi.org/10.5626/JCSE.2010.4.4.253`
11. Dusseault L, Snell J (2010) RFC5789: PATCH Method for HTTP. Tech. rep., IETF, URL `http://tools.ietf.org/html/rfc5789`
12. Fielding R, Reschke J (2014) RFC7231: Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content. Tech. rep., IETF
13. Fielding RT (2000) Architectural Styles and the Design of Network-based Software Architectures. PhD thesis, University of California, Irvine, URL `https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm`
14. Foster I (2005) Service-oriented science. Science (New York, NY) 308(5723):814–7, DOI 10.1126/science.1110411, URL `http://dx.doi.org/10.1126/science.1110411`

15. Fouilland F, Frasca M, Pelagatti P (1995) Monte Casasia (Ragusa). Campagne di scavo 1966, 197273 nella necropoli indigena. Notizie degli scavi di antichità 5-6:323–583, URL http://www.worldcat.org/oclc/758533779
16. Garshol LM (2004) Metadata? Thesauri? Taxonomies? Topic Maps! Making Sense of it all. Journal of Information Science 30(4):378–391, DOI 10.1177/0165551504045856, URL http://dx.doi.org/10.1177/0165551504045856
17. Haak LL, Fenner M, Paglione L, Pentz E, Ratner H (2012) ORCID: a system to uniquely identify researchers. Learned Publishing 25(4):259–264, DOI 10.1087/20120404, URL http://dx.doi.org/10.1087/20120404
18. Hill L, Koch T (2001) Networked Knowledge Organization Systems: introduction to a special issue. Journal of Digital Information 1(8), URL https://journals.tdl.org/jodi/index.php/jodi/article/view/32/33
19. Hjø rland B (2007) Semantics and Knowledge Organization. Annual Review of Information Science and Technology 41:367–405, DOI 10.1002/aris.2007.1440410115, URL http://doi.wiley.com/10.1002/aris.2007.1440410115
20. Hobbs JR, Pan F (2006) Time Ontology in OWL. report, W3C, URL http://www.w3.org/TR/owl-time/
21. Jones M, Hardt D (2012) RFC6750: The OAuth 2.0 Authorization Framework: Bearer Token Usage. Tech. rep., IETF, URL https://tools.ietf.org/html/rfc6750
22. Kunze J, Rodgers R (2013) The ARK Identifier Scheme. Tech. rep., Internet Engineering Task Force, URL https://tools.ietf.org/html/draft-kunze-ark-18
23. Kunze JA (2003) Towards Electronic Persistence Using ARK Identifiers. In: IWAW/ECDL Annual Workshop Proceedings 3rd, URL http://bibnum.bnf.fr/ecdl/2003/proceedings.php?f=kunze
24. Lorrio AJ, Zapatero GR (2005) The Celts in Iberia: An Overview. e-Keltoi 6:167–254, URL http://www.uwm.edu/celtic/ekeltoi/volumes/vol6/6_4/lorrio_zapatero_6_4.html
25. Miles A, Bechhofer S (2009) SKOS Simple Knowledge Organization System Reference. Tech. rep., W3C, URL http://www.w3.org/TR/skos-reference/
26. Page KR, De Roure DC, Martinez K (2011) REST and Linked Data. In: Proceedings of the Second International Workshop on RESTful Design - WS-REST '11, ACM Press, New York, New York, USA, p 22, DOI 10.1145/1967428.1967435, URL http://dl.acm.org/citation.cfm?id=1967428.1967435
27. Rabinowitz A (2012) GeoDia: or, Navigating Archaeological Time and Space in an American College Classroom. In: CAA2012 Proceedings of the 40th Conference in Computer Applications and Quantitative Methods in Archaeology, Southampton, UK, pp 259–268, URL http://dare.uva.nl/document/516092#page=260
28. Ren M, Lyytinen KJ (2008) Building Enterprise Architecture Agility and Sustenance with SOA. Communications of the Association for Information Systems 22, URL http://aisel.aisnet.org/cais/vol22/iss1/4/
29. Schulte WR (1996) "Service Oriented" Architectures, Part 2. Tech. rep., Gartner, URL https://www.gartner.com/doc/302869/service-oriented-architectures-
30. Schulte WR, Natis YV (1996) "Service Oriented" Architectures, Part 1. Tech. rep., Gartner, URL https://www.gartner.com/doc/302868/service-oriented-architectures-
31. Shaw R, Golden P (2013) Taking entity reconciliation offline. Proceedings of the American Society for Information Science and Technology 50(1):1–4, DOI 10.1002/meet.14505001107, URL http://doi.wiley.com/10.1002/meet.14505001107
32. Sporny M, Longley D, Kellogg G, Lanthaler M, Lindström N (2014) JSON-LD 1.0: A JSON-based Serialization for Linked Data. Tech. rep., W3C, URL http://www.w3.org/TR/json-ld/
33. Starr J, Willett P, Federer L, Horning C, Bergstrom M (2012) A Collaborative Framework for Data Management Services: The Experience of the University of California. Journal of eScience Librarianship 1(2):109–114, DOI 10.7191/jeslib.2012.1014, URL http://escholarship.umassmed.edu/jeslib/vol1/iss2/7
34. Stokstad M, Cothren MW (2013) Art history, 5th edn. Pearson, Boston
35. The Open Group (2013) Service Oriented Architecture: What Is SOA? URL http://www.opengroup.org/soa/source-book/soa/soa.htm
36. Tudhope D, Koch T (2004) New Applications of Knowledge Organization Systems: introduction to a special issue. Journal of Digital Information 4(4), URL https://journals.tdl.org/jodi/index.php/jodi/issue/view/20
37. Verborgh R (2014) DBpedia now available as triple pattern fragments. URL http://lists.w3.org/Archives/Public/public-lod/2014Oct/0293.html
38. Zeng ML (2014) NKOS (Networked Knowledge Organization Systems). URL http://nkos.slis.kent.edu/