Exploiting User Tags to Build a Semantic Cultural Heritage Portal

Kees van der Sluijs, *Eindhoven University of Technology*

Geert-Jan Houben, *Delft University of Technology*

Based on a metadata structure that intelligently builds a semantically linked data set, the Chi Explorer Web application discloses cultural heritage collections to the general public.

The Regional Historic Center Eindhoven (RHCe, www.rhc-eindhoven.nl) governs all official city archives and the cultural historic information related to Eindhoven and its surrounding municipalities. The RHCe archives contain millions of objects, including official documents, pictures, videos, maps, and newspapers. These objects are organized in collections based on their function and type.

One of RHCe's missions is to disclose their collections to the general public. The current practice is to physically retrieve the objects for interested visitors, based on handcrafted metadata indices constructed by RHCe's professional annotators. This is time-consuming for visitors and RHCe staff. Many objects are also fragile, and physical contact could be damaging. To address these problems, RHCe has begun creating high-quality digital copies of their source data for future information disclosure. They envision their archives reaching a larger public over the Web and thus getting a better idea of the evolving interests and wishes of their visitors and users. An important element in this process is metadata. Currently, RHCe employs professionals to correctly annotate objects, but the amount of new objects grows much faster than the annotators can keep up with, so its huge archives are only partially annotated. We could greatly lessen this burden, however, by getting the users to help with annotation.

This article shares the experience of using semantic techniques to create an overall metadata structure that intelligently defines a semantically linked heterogeneous data set. Based on this data set, we have been able to offer a navigation structure over the archive objects and study how to exploit the metadata in the navigation. Because involving user groups was an important goal in this process, we use intelligent techniques to help convert unstructured user-generated metadata (obtained with easy-to-use interfaces) to well-structured ontology-based metadata ready for the professional annotators. Thus, we illustrate how to apply intelligent techniques for metadata integration and user participation in the practice of small- and medium-sized cultural heritage institutions.

1541-1672/10/$26.00 © 2010 IEEE
Published by the IEEE Computer Society

Semantic Integration

One of our first challenges was integration. Previously, all metadata information was stored in separate proprietary database applications for legacy reasons.

The desire was to have one Web application that let users browse the RHCe data sets as if it was one integrated homogeneous data set. We therefore first needed to create an integrated data model that describes all metadata fields from all metadata databases. We were especially keen on identifying the metadata fields that the separate collections have in common because RHCe saw great value in connecting objects in different collections. For example, someone might be interested in a marriage certificate (for example, his grandparents') and then want to see photos of the church from the time of the wedding.

It is important to point out that considerations like these were inspired by the added value that the new interface could provide. That is also why we have presented these aspects as part of the complete study. An advantage of the integrated data set approach is that it would let us create specialized visualization primitives for navigation. Furthermore, it also let us use it as a format for user-generated metadata.

When we constructed the overall metadata structure, we needed to select the formalism. Basic requirements were that it should be easily extendible and that it should be possible to easily integrate or connect new data sources, such as additional legacy sources from mergers and data sources from other institutions. Furthermore, based on our work planned on navigation primitives, we decided to connect our metadata source to external knowledge sources. We chose the Resource Description Framework (RDF) because it fits our requirements. RDF is a flexible metadata language, easily extendible and as a Web standard allows for exchange of data between sources.

Before translating the various databases to RDF, we first designed a schema for our combined data set. We needed to combine the given properties of the available sources and standards (or lack thereof) and the desired properties of the solution RHCe desired.

We wanted to base our schema on open standards (vocabularies) where possible to improve extendibility and reuse. We used a Dutch standard called Gabos for describing the topographic historic information in our collection (developed by a collective of many local archives and local cultural centers in the Netherlands). Gabos however only provided a flat list of descriptors with proper names to describe objects. We mapped this content descriptor structure to RDF.

The desire was to have one Web application that let users browse the RHCe data sets as if it was one integrated homogeneous data set.

We also designed specialized faceted browsing visualizations (which we describe later on). To make this possible, we used the structure of a specialized ontology for each facet: location, domain, and time ontologies. These ontologies are also important in the user-generated metadata phase; there we exploit their labels and semantic structure.

Both the location and domain ontology are based on data structures developed internally in RHCe. The location ontology relies on a hierarchy of locations and their coordinates. The locations have been extended with time properties where appropriate that can be used in the historical perspective to support evolving locations, such as with city restructurings. The domain ontology was based on a classification hierarchy developed for structured annotation in the various RHCe database applications. In both ontologies, we reused standardized relationships where possible, for example, for hierarchical relationships showing the broader and narrower relationships from the Simple Knowledge Organization System (SKOS) vocabulary (www.w3.org/2004/02/skos).

For the time dimension, RHCe didn't have a structured ontology yet, even though different time notions are often used in their metadata databases. For this important notion in cultural heritage applications, we reused the OWL Time ontology and mapped all time notations used in the metadata databases to instances of that ontology.

The next step was to make wrappers that can translate the metadata information from the existing database applications in RHCe into instances of our RDF schema. We created unique URIs for objects based on their database ID and translated all metadata in the databases into terms of our metadata structure with relations to the relevant ontologies. Mapping the actual data items to instances of our schema was a relatively straightforward process, besides some restructuring issues. The time notations could be easily translated to OWL Time notations. The domain ontology link was also rather

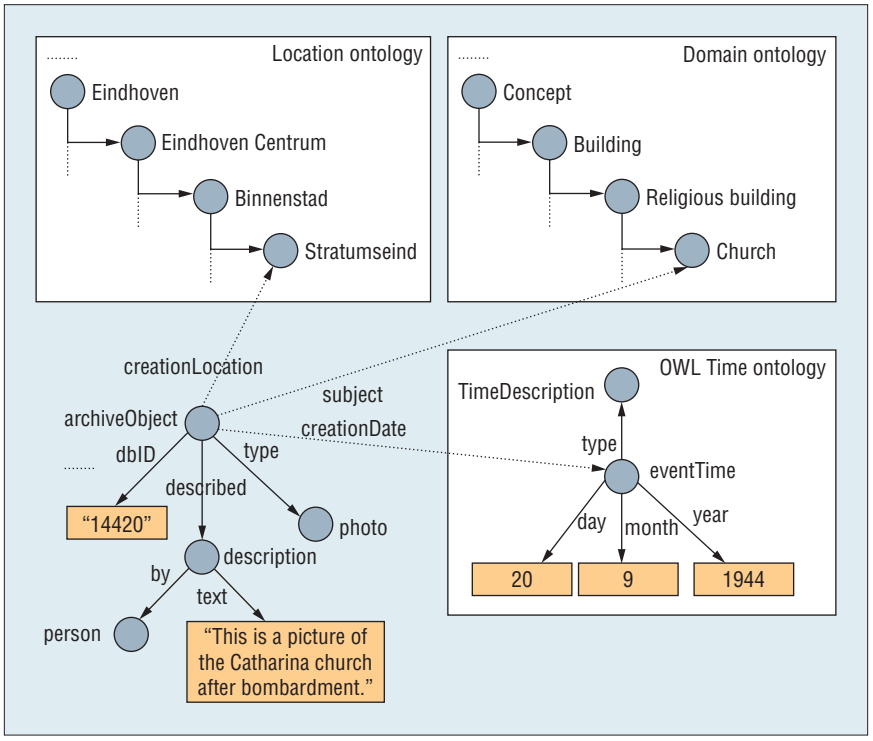

Figure 1. Chi metadata structure. The metadata describes a picture of the Catharina church after its bombardment on 19 September 1944, with the added semantic links between the picture and the ontologies.

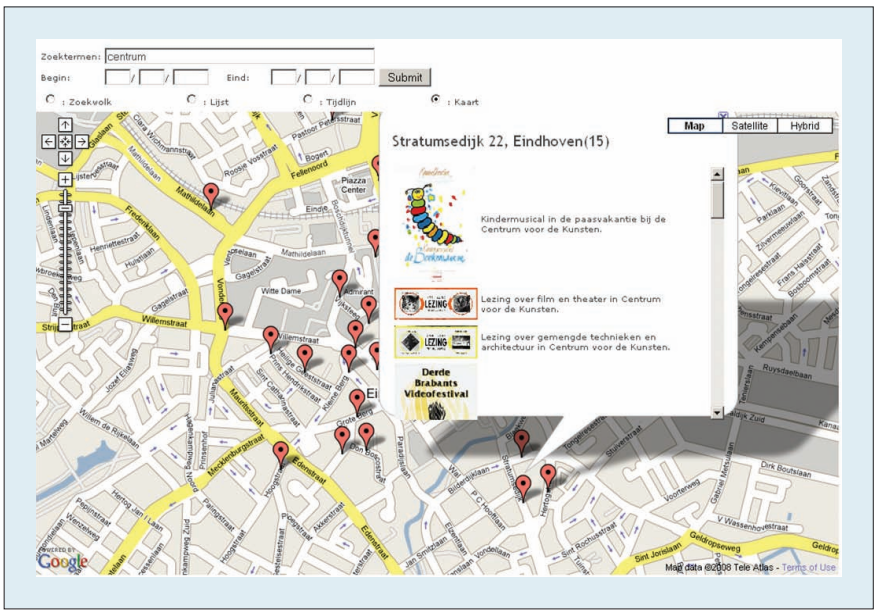

Figure 2. Chi Explorer map view. We used the added links between object locations and Google Maps to visualize locations of objects.

straightforward as long as the annotators correctly used the classification hierarchy. We only considered the classifications that exactly matched the classification hierarchy. For the location hierarchy, we had to do some more elaborate matching and mapping, for which we used the Relco framework, which we explain in more detail later.

Figure 1 gives an example of an instance (simplified) in our metadata structure. We created the links with the ontologies.

Navigation and Visualization

Based on this integrated data set, we built the Chi framework. We first looked at how we could build the user interface for navigation and search.

The RHCe user base is typically interested in objects related to specific areas, such as our previous example in which a user is interested in the location of his grandparents' marriage. Therefore, we added a map view, which should simplify navigating objects over their spatial relationships. We used the Google Maps API to visualize object location. Figure 2 shows a screenshot from the Chi Explorer map view. Here we were able to exploit that in our integrated data set we transformed location annotations to links in our location ontology. This ontology contains location coordinates that are translated to Google Maps coordinates.

A search or browsing action can lead to the need to visualize a large set of objects on a relatively small part of the map (most are within Eindhoven). Our user study (which we describe later) showed that users struggled with large unordered sets of objects. Therefore, we incorporated a clustering algorithm based on the Hierarchical Agglomerative Clustering Algorithm.[1] However, instead of just finding a clustering for a group of points, we reduce the number of visualization points by clustering until we have reached the maximum. The algorithm works as follows:

1. Define the maximum number of clusters that will be visualized on the map.

2. Compute the proximity matrix containing the distance between each pair of search results. Treat each search result as a cluster.
3. Find the closest pair of clusters using the proximity matrix. Merge these two clusters into one cluster. Update the proximity matrix to reflect this merge operation.
4. Stop when the number of clusters equals the maximum.

The clustered sets of objects are afterwards visualized in the Map view. When the user selects a cluster, the set of objects associated with that location is visualized. For every object, a thumbnail representation is shown together with a description. Within a cluster of objects, the objects can optionally be grouped by creating subsets that have similar properties in one of the other dimensions—that is, time and domain concepts. In this way, we can group objects in an object set either on the time periods or on the concepts for a certain location, which makes browsing an object set more orderly.

Similarly, the user base is typically interested in objects from a specific time period. For example, a user might be interested in objects from the time his grandparents married. Therefore, we added a timeline view in which objects can be related by the time periods in which they were created. We used a Simile timeline component (see http://code.google.com/p/simile-widgets) for that. Here we exploited the fact that all date notations in our integrated data set were translated into OWL Time instances. Thus, we can generate dates and intervals as elements on the timeline. Dates at different granularity levels can be recognized in the view by the different element sizes.

For objects on the timeline, we faced a similar problem of cluttering the timeline with too many objects. Objects on the timeline with overlap in the time period are arranged vertically. We had to make sure that the number of vertically placed events fit the screen size. For this, we use a slightly adapted version of the clustering algorithm:

1. Determine the candidate time frames with more results than the maximum.
2. Create a proximity matrix containing the time-distance between each pair of search results in a candidate time frame. Treat each search result as a cluster. (We measure the distance between time frames by adding the amount of time necessary to add to both time frames to come to a union of those time frames. For instance, given the time frames "1940–1945" and "5 May 1943," the union of those time frames is "1940–1945" and it takes no time to add to the first time frame and five years minus a day to the second; the total distance between the two is five years minus a day.)
3. Find the closest pair of clusters using the proximity matrix. Merge these two clusters into one. Update the matrix to reflect this merge.
4. Stop when the number of clusters equals the maximum.
5. Repeat steps 3 through 5 for each candidate time frame.

The Chi Explorer Web-based application lets RHCe visitors efficiently browse the archives based on structured annotations.

Like in the map facet, a visualization of the objects in an object set can be grouped in the other dimensions (location and domain concepts).

The last visualization we considered was based on many users' desire to quickly navigate objects that relate to a specific theme or concept. In our example, the user is interested in objects that relate to the concept of church or marriage. The domain ontology was in fact specifically designed to facilitate this thematic classification of objects, which lets users navigate semantically related objects. For visualization of the domain ontology, we used a graph-based visualization based on the GraphViz graph library (www.graphviz.org). We directly visualize the domain ontology, and using that, our integrated data set links data objects to the concept in this ontology.

We built the Chi framework to facilitate the user interface we just described (see Figure 3). The Chi Engine is the application back end that allows storing and querying the data. Furthermore, it maintains a user model where user information and user-generated input can be stored. Querying and storing RDF data in the Chi Engine is based on Sesame (www.openrdf.org). The Chi Engine also uses a component that provides suggestions for mappings between user tags and concepts in the ontologies. The top part of Figure 3 contains the presentation logic, Chi Explorer, which provides the specialized user interface over the data from Chi Engine.

The faceted navigation depends on the relations between our integrated data set and the ontologies. Given the lack of metadata for many objects, we use the Relco component to allow

user-generated metadata, including links to the ontologies, which makes the navigation via our visualizations possible.

User-Generated Metadata

The Chi Explorer Web-based application lets RHCe visitors efficiently browse the archives based on structured annotations. However, the center has currently only rich annotations available for a relatively small part of its continuously growing archive. Therefore, as part of their new process for creating the metadata, it was decided to attempt using the knowledge and involvement of the visitors to augment the level of well-structured annotations.

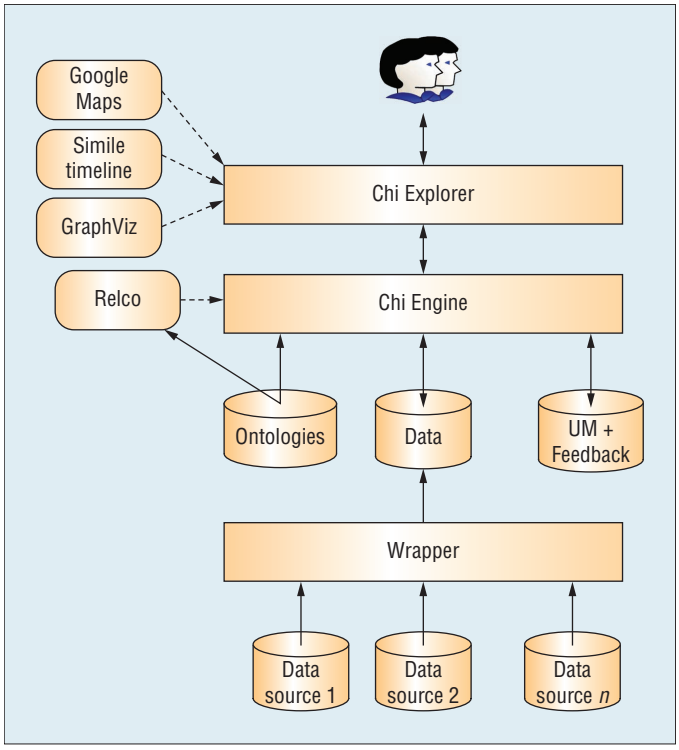

Figure 3. Chi framework architecture. The bottom shows the RHCe back ends (the proprietary databases) and the wrapper that transforms that data to instances of our integrated structure. The Chi Explorer provides the specialized user interface over the data from Chi Engine.

To accommodate lay users, we keep the annotation process simple by extending the system with a tag-based interface.[2,3] To guarantee the quality of the system's metadata, it was decided to maintain the current data structures based on RHCe's internal ontologies as the basis for navigating the archives, also because of the known difficulties reported for using tag-only approaches.[2] Therefore, we built a software tool named Relco that relates user tags to the concepts in the ontologies. Relco depends on three properties of the ontologies to which we try to relate—namely, the availability of textual representations of concepts (such as labels), meaningful semantic relationships between concepts, and enough concepts to cover most of the object domain. We fulfilled these requirements with the ontologies we described earlier.

Our work differs from related work that tries to build ontologies based on tags (folksonomies).[3] The Relco approach to relate a tag to concepts is partially inspired by work on ontology matching. It follows the frequently used method in ontology matching[4,5] of finding a set of concepts with the highest syntactic and semantic similarity between the input and the target ontologies. However, our situation is slightly different than other systems, so we focused on extending regular techniques with techniques that exploit our data structures and system users.

Step 1: Lexical Matching

The first step Relco takes is finding syntactic matches between tags and concepts—for example, matching the user tag "church" and the concept in our ontology labeled "church."

For this, first the textual representation for a concept in the given domain ontology must be determined. Because identifying the appropriate textual representation for concepts is ontology-dependent and Relco is designed as a generic tool, we made it possible for it to cover most practical situations.

Usually concepts contain a property such as `rdfs:label` to denote a given concept's label. Sometimes, there are multiple candidate labels for a concept. For instance, the SKOS vocabulary uses the `skos:prefLabel` and `skos:altLabel` properties to distinguish preferred labels and alternative labels. Multilabels also occur often in ontological sources that support multiple languages. We also see sources that use a more complex naming schema with intermediate label nodes. Lastly, we consider labels that are encoded in URIs, usually as part of the fragment identifier. The Relco configuration supports each of these labeling scenarios via different constructs—for the simple cases only the appropriate properties must be specified, for the more complex cases a modified SPARQL query (see http://www.w3.org/TR/rdf-sparql-query/) can be used, and for URI-encoded labels the delimiter characters have to be specified.

After having thus obtained the labels for all the concepts in the source ontology, we can match the input tag to those labels. To do that, we first stem both the tag and labels (using Snowball, http://snowball.tartarus.org) to account for morphological

differences between words. Because user tagging is typically a quick-and-dirty process that frequently includes spelling errors and words spelled in several different ways (we observed this especially with names in this application), we chose to include some robustness and flexibility in the comparison of tags with labels. To this end, we use an algorithm for computing the similarity between strings. While many similarity metrics are available, in Relco we use the open source Simmetrics library (http://sourceforge.net/projects/simmetrics) for our lexical matching needs because it contains implementations for many of the well-known string similarity algorithms.

A straightforward use of one of these similarity metrics algorithms however has one big drawback: it does not scale well because every tag a user suggests would have to be compared with every label in the ontology. Tests on a reasonably large representative ontology with approximately 3,800 concepts showed it would take about seven seconds on a standard modern computer to match the input tag with the string representations of those concepts.

To improve on this performance, Relco uses Apache Lucene (http://lucene.apache.org) to build a fuzzy index. The idea is to compute *n*-grams of words, meaning that strings are broken up into all subsequences of length *n*. These subsequences are put in an inverted index. Using the heuristic that two only slightly differing strings share many common subsequences, an input term's *n*-grams can be quickly compared with the labels in the ontology. The result of this process is a set of matching labels with a similarity measure that represents the number of matching *n*-grams between the input tag and matching label. However, experiments showed that the accuracy of this similarity measure is in general worse than that of the previously discussed algorithms. Therefore, we use Lucene with a relatively low accuracy setting to quickly preselect all candidate strings that might match. We then use the string similarity algorithms from the Simmetrics library for higher-quality string comparisons. Using this method on the same test ontology with 3,800 concepts reduced the computation time to less than a second.

The result of the lexical matching process in Relco is a set of candidate concepts of which the label (or one of them) matches the input tag, together with a certainty value [0..1] that represents the similarity measure. This set is ordered by certainty, and a certainty threshold specifies the minimum similarity measure to consider two strings a match.

> All the users became enthusiastic about the possibilities with the new interfaces after they used them for some time.

Step 2: Exploiting Semantic Structure

The result of Relco's first step is a set of concepts with labels that are syntactically related to the input tag. Next, Relco exploits the given ontologies' semantic structure to extend that result set with semantically related concepts. Suppose, for example, that a user uses the input label "religious building." Instead of only matching to the concept "religious building," it can also offer the user more specific types of those buildings, such as a "chapel," "church," "mosque," or "temple."

A given ontology represents the (typed) relations between the concepts. "Following" these relationships between concepts, we find semantically related concepts that might also be relevant for the input tag, even though they syntactically might be completely unrelated. For this purpose, some relationships in the ontology might be more relevant than others. For example, with the properties "related" and "antonym" available in the ontology, it might be better to only exploit the "related" property in this step. We can configure this in Relco as well as the depth of the semantic expansion. For example, it might be useful to follow a property more than once for some sources, finding related concepts at a greater distance from the original concept.

It is also possible to use more complex relationships between concepts—that is, relationships that cannot be found by following a single property, but only by using more complex paths between concepts. For example, if the user inputs "temple" as a tag, we might not only want to return concepts for more specific temples but also sibling concepts like "chapel," "church," or "mosque" in case the user made a semantic mistake in his input tag. To retrieve sibling concepts, we need to follow the "broader" property first to find the more general concept (such as "religious building") and then follow the "narrower" property to find its more specific concepts. To accommodate such complex relationships, Relco can be configured by defining SPARQL queries.

Table 1. Example tag suggestions provided by Relco (in Dutch).

Input	Suggestions	Input	Suggestions
Christendom	Christenen	Zang	Muziek
	Religie		Liederen
	Katholicisme		Karaoke
	Oosterse kerken		Koor
	Protestantisme		Musical
	Aartsbisschop		Opera
	Bisschop		Operette
	Christelijk onderwijs		Songfestival
	Kerk		Zand
	Priester		Geologie
	Zonde		Grondsoorten
	Koning		Strand

In this step, we also determine the certainty for the new concepts. The certainty values for the new concepts are based on the certainty values of the original concepts obtained in step 1, but we lower them by a configurable reduction factor.

Step 3: Context Disambiguation

For some input tags, Relco might come up with many related concepts. These related concepts are already ranked on similarity, but by exploiting the properties of our tagging system, we can do more. Previous tags might contain hints that can help disambiguate and better select the most appropriate concepts. For example, if a user inputs the tag "religious building" and we already have a previous tag "Christian," the suggestions "chapel" and "church" could be regarded more relevant than "mosque" or "temple." The underlying assumption is that some of these previous tags can provide insights into what the user meant.

The disambiguation technique is based on the ontology's structure. If we consider the concepts that relate closely to each other via a maximum number of relationships as a neighborhood, the probability is high that some of the tags that describe an object will relate to concepts in the same neighborhood. In our example ontology, there is the "related" property between "Christian" and "church" and between "Christian" and "chapel." The path from "Christian" to "mosque" is much longer, and therefore, "mosque" is considered not to be in the neighborhood of "Christian."

To perform this disambiguation efficiently, we preprocess the neighborhoods of the ontology's concepts C. Initially, a $|C| \times |C|$ distance matrix is built and the part of the matrix with distances smaller or equal to a constant n is stored. This n is typically small, especially for well-connected graphs, but a good value depends on the ontology's structure.

When a previous tag and the current tag have concepts in the same group—that is, if the distance between the concepts is below a predefined threshold—the certainty for that concept is increased with a configurable factor. As a result of step 3, the certainty values for the result set are altered regarding the context.

Table 1 contains two examples of input tags and Relco suggestions (in Dutch). In the first example *Christendom* (Christianity) is related to Christian concepts, except maybe the less obvious connection with *koning* (king). In the second example *Zang* (singing) is related to music concepts, but also to *zand* (sand) because of the close syntactical resemblance between *zang* and *zand*.

User Study

The goal of our first user study was to see if users appreciate the semantic navigation interfaces in Chi Explorer. The study targeted RHCe's three main user groups:

1. RHCe's largest user group are senior citizens. They are generally interested in genealogy (usually their own family) and everything they relate to their childhood.
2. Another substantial user group, and one that RHCe would like to grow, is students. RHCe has contacts with different schools in the neighborhood.
3. Historians make up the a third group that is smaller in size, but important for RHCe. They use the archives to study Eindhoven's history and need metadata to search through the data.

We used five users from each of RHCe's three main user groups: external young users (between 15 and 25 years old), external senior users (between 55 and 65 years old), and RHCe internal users (historians).

The 15 participants were given a list with 12 assignments to find and navigate objects available in the system. An example task is, "Find photos of the center of Eindhoven that feature buildings damaged in the Second World War." We also advised them to try using all the available facet visualizations. During these tasks, we let the users think aloud and filmed their reactions. At the end, they also filled in a questionnaire.

We found that it took all the users time to get used to the visualization interfaces, especially for the Graph visualization, even the RHCe internal

users that recognized the concept descriptor structure. We also saw differences between the groups: the younger users grasped the new visualizations more easily than the other groups. For example, most of the younger ones had used the Maps visualization before, while some senior users overlooked the zoom functionality. However, all the users became enthusiastic about the possibilities with the new interfaces after they used them for some time.

We also observed conflicting feature requests between the groups. For example, the internal users preferred to see a larger part of the concept graph in the Graph view because they were already familiar with the graph and wanted to navigate faster. The other two groups however strongly preferred the existing Graph view because they felt it became too complex if extended. Here we see an opportunity for personalization.

One result of the study was that users struggled with large result sets within a specific facet. Based on this feedback, we introduced the clustering we described in this article, which we did not have yet during the study. Another issue that we are currently looking into is combining facets for searching because users would like to select a region with objects in the Map view and then navigate through the timeline for these objects.

Interest and research in using Semantic Web techniques in the cultural heritage domain has been flourishing lately.[6] This research shows that semantic Web techniques can be used in several additional ways besides the one that we discussed in this article. For example, there is progress in extracting metadata from natural language descriptions,[7] using encyclopedias to create ontological knowledge sources,[8] and analyzing materials and techniques in artworks to disseminate the collection to different types of users.[9] And these are only some of the subjects that have been explored.

The Chi Explorer navigation builds on related work. For instance, mSpace[10] and /facet[11] present faceted browsers. These are more general and more flexible than Chi Explorer because they can use any RDF property as basis for a facet. Their approach supports a powerful, but easy query paradigm, which is something we would like to incorporate in a future Chi Explorer iteration as well. Choosing some well-defined facets obviously means that we can build specialized visualizations for them, like we did with Google Maps, the Simile timeline, and Graphviz. /facet actually supports specialized facet visualizations as well, and Museum Finland did something similar in the recent Culture Sampo Semantic Web 2.0 portal for cultural heritage.[12] Culture Sampo is a much more extensive Web application than Chi Explorer. For visualization, it uses the same visualizations as Chi Explorer (however in more versatile ways), even though it does not incorporate a clustering algorithm. The main difference with our work is that we focused on user-based input for not yet well-annotated collections, where they used already well-structured data sets.

Although the RHCe project is smaller in scale than the ones we have just described, our results show that the semantic techniques we use are applicable and useable for small- and medium-sized parties like the regional historic centers in the Netherlands. The discussed techniques solve issues that organizations like RHCe have been struggling with for a long time.

Our framework lets us innovatively connect to the user base, and it provides a serious opportunity to expand that base. The mechanisms we built for user interaction give RHCe a new communication channel with their users. This reduces overhead costs as it reinvents the metadata creation process to face the continuous expansion of collections.

The integrated data model and ontology intelligently forms a semantically linked heterogeneous data set that represents an integrated view over the collections. Besides the subsequent advantage for users in their navigation, it has become the main tool to reason about the combined data. It has lifted the identification of relations between collections to a higher level, resulting in better support for (professional) cross-collection searches and for richer annotations. In particular, the role of semantic languages has brought this integration process an important step forward and has created

THE AUTHORS

Kees van der Sluijs is a PhD student in computer science at the Eindhoven University of Technology and the project manager of the Grapple European research project, which deals with Adaptive Learning Environments. His research is part of the Hera research program that combines research on Web information systems. Van der Sluijs has an MSc in computer science from Eindhoven University of Technology. Contact him at k.a.m.sluijs@tue.nl.

Geert-Jan Houben is a professor in Web information systems at the Delft University of Technology. In Delft and Eindhoven, he leads the Hera research program on Web Information Systems. His group concentrates in its research on large-scale information systems, specifically information systems that involve Web and Semantic Web technology. Houben has a PhD in computer science from Eindhoven University of Technology. Contact him at g.j.p.m.houben@tudelft.nl.

a new instrument to effectively cope with the constantly changing and expanding collections.

References

1. A.K. Jain, N.M. Murty, and P.J., Flynn, "Data Clustering: A Review," *ACM Computing Surveys*, vol. 31, no. 3, 1999, pp. 264–323.
2. S.O. Choi and A.K. Lui, "Web Information Retrieval in Collaborative Tagging Systems," *Proc. Int'l Conf. Web Intelligence*, IEEE Press, 2006, pp. 352–355.
3. L. Specia and E. Motta, "Integrating Folksomonies with the Semantic Web," *The Semantic Web: Research and Applications*, Springer, 2007, pp. 624–639.
4. Z. Aleksovski, W. ten Kate, and F. van Harmelen, "Ontology Matching Using Comprehensive Ontology as Background Knowledge," *Proc. Int'l Workshop Ontology Matching at ISWC 2006*, 2006, CEUR, pp. 13–24.
5. E. Rahm and P.A. Bernstein, "A Survey of Approaches to Automatic Schema Matching," *VLDB J.*, vol. 10, no. 4, 2001, pp. 334–350.
6. L. Hardman et al., "Using AI to Access and Experience Cultural Heritage," *IEEE Intelligent Systems*, vol. 24, no. 2, 2009, pp. 23–25.
7. T. Ruotsalo, L. Aroyo, and G Schreiber, "Knowledge-Based Linguistic Annotation of Digital Cultural Heritage Collections," *IEEE Intelligent Systems*, vol. 24, no. 2, 2009, pp. 64–75.
8. R. Witte et al., "Converting a Historical Architecture Encyclopedia into a Semantic Knowledge Base," *IEEE Intelligent Systems*, vol. 25, no. 1, 2010, pp. 58–67.
9. G. Karagiannis, "Using Signal Processing and Semantic Web Technologies to Analyze Byzantine Iconography," *IEEE Intelligent Systems*, vol. 24, no. 3, 2009, pp. 73–81.
10. M.C. Schraefel et al., "The evolving mSpace Platform: Leveraging the Semantic Web on the Trail of the Memex," *Proc. 16th ACM Conf. Hypertext and Hypermedia*, ACM Press, 2005, pp. 174–183.
11. M. Hildebrand, J. van Ossenbruggen, and L. Hardman, "/facet: A Browser for Heterogeneous Semantic Web Repositories," *Proc. Int'l Semantic Web Conf.*, Springer, 2006, pp. 272–285.
12. E. Hyvönen et al., "CultureSampo: A Collective Memory of Finnish Cultural Heritage on the Semantic Web 2.0," Helsinki Univ. of Technology and Univ. of Helsinki, 2008.

Selected CS articles and columns are also available for free at http://ComputingNow.computer.org.