Accessibility Commons: A Metadata Infrastructure for Web Accessibility

Shinya Kawanaka[†]
shinyak@jp.ibm.com

Yevgen Borodin[‡]
borodin@cs.sunysb.edu

Jeffrey P. Bigham[#]
jbigham@cs.washington.edu

Darren Lunn
darren.lunn@cs.manchester.ac.uk

Hironobu Takagi[†]
takagih@jp.ibm.com

Chieko Asakawa[†]
chie@jp.ibm.com

[†]IBM Tokyo Research Laboratory, 1623-14 Shimotsuruma, Yamato, Kanagawa, 242-8502, Japan
[‡]Dept. of Computer Science, Stony Brook University, Stony Brook, NY 11794, USA
[#]Department of Computer Science and Engineering, University of Washington, Seattle, WA 98195 USA
◦ School of Computer Science, University of Manchester, Kilburn Building, Oxford Road, Manchester, M13 9PL, UK

ABSTRACT
Research projects, assistive technology, and individuals all create metadata in order to improve Web accessibility for visually impaired users. However, since these projects are disconnected from one another, this metadata is isolated in separate tools, stored in disparate repositories, and represented in incompatible formats. Web accessibility could be greatly improved if these individual contributions were merged. An integration method will serve as the bridge between future academic research projects and end users, enabling new technologies to reach end users more quickly. Therefore we introduce *Accessibility Commons,* a common infrastructure to integrate, store, and share metadata designed to improve Web accessibility. We explore existing tools to show how the metadata that they produce could be integrated into this common infrastructure, we present the design decisions made in order to help ensure that our common repository will remain relevant in the future as new metadata is developed, and we discuss how the common infrastructure component facilitates our broader social approach to improving accessibility.

Categories and Subject Descriptors
K.4.2 [**Social Issues**]: Assistive technologies for persons with disabilities; H.3.5 [**Online Information Services**]: Web-based services

General Terms
Design, Human Factors, Standardization

Keywords
Web accessibility, metadata, annotation, database

ASSETS'08, October 13–15, 2008, Halifax, Nova Scotia, Canada.

1. INTRODUCTION
Metadata, extra information added to original documents, has the potential to improve Web accessibility. For example, alternative text added to images is one of the simplest and most useful forms of metadata, and allows visually impaired people to understand the function of an image. The use of metadata in improving the Web accessibility is very broad. Here are just a few of the most common uses: heading tags for navigation, labels for form elements, and ARIA markup indicating the semantic roles of dynamic content.

In general, there are two types of metadata: internal metadata that is embedded in original documents and external (stand-off) metadata that is stored separately (but which is associated with the original documents). Internal metadata can only be authored with the appropriate permissions, while the external metadata can be created by anybody. The power of the external metadata is that people with different interests or requirements can act on their own without affecting the original content or requiring the content owners to be involved.

The main challenge in providing external metadata is in the on-the-fly association of specific metadata with the content it describes. Anchoring metadata to a specific part of a document is, therefore, the key to effective use of external metadata. Various research projects have focused on automatic or semi-automatic creation and adaptation of external metadata to leverage accessibility metadata, based on transcoding systems or self-voicing browsers.

However, since most projects are isolated efforts, the metadata is isolated in separate tools, stored in disparate repositories, and represented in incompatible formats. Web accessibility could be improved if the individual contributions were merged. For example, if one system can automatically infer alternative text and another project can automatically infer main content position, the combination may dramatically improve accessibility for visually impaired users. A method for integrating these disparate types of metadata will enable research institutes worldwide to develop a comprehensive sets of tools based on shared technology. Most importantly, a shared community infrastructure will serve as a bridge between future academic research projects and end users, enabling new technology to reach end users more quickly.

To this end, we propose a common metadata infrastructure, *Accessibility Commons (AC)*, whose goal is to integrate and share

the metadata produced by various research projects, assistive technologies, and individuals who are producing metadata. The AC seeks to enable this needed and ambitious goal through the following two contributions: (i) a flexible schema for representing a common metadata repository and (ii) a method for integrating metadata of disparate types.

In this paper, we first review existing research projects involving metadata and non-visual web-browsing, and then propose a common environment for metadata integration and sharing. We also discuss the design decisions that will help our common repository remain relevant in the future as new metadata is developed. We then describe several examples and scenarios of using a common metadata environment in aiBrowser, HearSay, and SADIe. Finally, we conclude by explaining how the Accessibility Commons can contribute to the larger goal of social accessibility.

2. RELATED WORK

The work related to the Accessibility Commons includes (i) numerous accessibility research projects and products that generate and use metadata, and (ii) projects in other domains that deal with the problem of data integration among applications.

2.1 Research Projects and Screen Readers

This section summarizes the metadata that is already in use by the existing accessibility research projects and products. A thorough understanding of the existing metadata helped to inform our decisions and strategy, and hopefully ensures the Accessibility Commons will remain relevant as new projects and products are developed.

2.1.1 aiBrowser

The aiBrowser [14] is a multimedia browser for visually impaired people. The browser transcodes HTML documents and Adobe Flash [3] on the client side to provide alternate content that is more accessible for visually impaired people. The transcoding is done using metadata described in XML. The metadata describes how to combine HTML elements and Flash objects to generate more accessible alternate content. In the metadata, XPath expressions are used to specify HTML elements and Flash queries are used to specify Flash objects. In addition, the aiBrowser allows users to add manual annotations for headings and alternative text. If the aiBrowser were to use a common repository, it could share its metadata and user annotations to provide alternative text and heading tags to people using other technologies.

2.1.2 HearSay

The HearSay non-visual Web browser [15] uses various content analysis techniques to improve Web accessibility. Among them are: context-directed browsing for identification of relevant information in webpages [18], language detection [16], concept detection [17], etc. HearSay uses the results of the automated analyses to annotate Web content. For example, the context-directed browsing algorithm inserts a "start" label, instructing the browser to begin reading the page from a specific position. The HearSay browser has a VoiceXML-based dialog interface, which interprets the labels and provides facilities for navigating, editing, and creating manual labels. The labels can be stored in personal or shared repositories. The use of uniform metadata and a shared repository allows other applications to benefit from the labels created in HearSay. At the same time, future HearSay users will have access to metadata created by a wider pool of blind Web users.

2.1.3 WebInSight for Images

WebInSight for Images [11] provides alternative text for many Web images to improve their accessibility. To make this alternative text, WebInSight uses contextual analysis of linked webpages, enhanced Optical Character Recognition (OCR), and human labeling. The alternative text strings are stored in a shared database referenced by an MD5 hash of the image and the URL of the image. The stored alternative text is supplied as users browse the Web. When a user visits a webpage for the first time, WebInSight attempts to create alternative texts by doing contextual analysis and OCR. If these options fail, the user can request human labeling. By combining the alternative text into a common database, users will be more likely to experience the benefits.

2.1.4 Site-wide Annotation

Site-wide Annotation [20] is a research project to transcode entire websites by annotating them. The metadata of the site-wide Annotation uses XPath expressions. The system checks for elements matching the expressions and transcodes the webpages based on the metadata. This allows transcoding an entire website with a small set of metadata. If this metadata can be created and shared by users, a larger number of websites could be transcoded for better Web accessibility.

2.1.5 Social Accessibility

Social Accessibility [19] is a project to improve Web accessibility by providing a collaborative metadata authoring mechanism that combines social computing techniques and accessibility technologies. The approach can drastically shorten the time for accessibility renovations, since any volunteer user can fix the problems per users' requests. When users find problems, they can report them from their familiar browsing environment. The requests are sent to a social accessibility server and collaboratively fixed by the volunteers. Initially, it supports alternative text and heading tags that are stored in the common repository. It also provides a client interface to seamlessly apply the metadata to webpages. It is a proof of concept for our proposed common environment.

2.1.6 AxsJAX

AxsJAX [4] is an accessibility framework to inject accessibility support into Web 2.0 applications. Currently the main targets of AxsJAX are Google applications such as GMail and Google Docs. AxsJAX scripts use Greasemonkey, a bookmarklet, or run directly in Fire Vox [2], a screen reader implemented as a Firefox Extension. AxsJAX identifies elements to which ARIA markup should be provided using XPath. Currently, these associations are distributed to users in the form of scripts. More tools could benefit from the semantic knowledge encoded in these scripts if they were stored in a more flexible and semantically-accessible common repository.

2.1.7 Accessmonkey

Accessmonkey [12] is another common scripting framework that Web users and developers can use to collaboratively improve Web accessibility. The goal is to enable both Web users and developers to write scripts that can then be used to improve the accessibility of webpages for blind Web users. An example script provided by the Accessmonkey demonstrates how WebInSight for

images can be implemented in this framework in order to provide alternative text for Web users as they browse and to suggest alternative text for Web developers trying to improve their pages.

2.1.8 Structural Semantics for Accessibility and Device Independence (SADIe)

SADIe [21] is a proxy-based tool for transcoding entire websites as opposed to individual pages. It relies on ontological annotations of the Cascading Style Sheet (CSS) to broadly apply accurate and scalable transcoding algorithms. Only by explicitly enunciating the implicit semantics of the visual page structure (groups, components, typographic cues, etc.) can we enable machine understanding of the designers' original intentions. These intentions are important if we wish to provide a similar experience to visually impaired users as to fully sighted users. SADIe can be regarded as a tool for the site-wide reverse engineering of webpages to achieve design rediscovery [22].

2.1.9 JAWS

JAWS is one of the most popular screen readers. It has a labeling feature, which allows users to provide alternative text for images or flash objects. The latest version of JAWS can make use of WAI ARIA [10], a World Wide Web Consortium (W3C) internal metadata standard, to improve the accessibility of dynamic content.

2.1.10 Summary

The common elements of the metadata mentioned in the previous section are the following four elements: URI addressing, element addressing, conditions, and semantics (the meaning of the metadata). We will explain how these kinds of metadata can be captured in our database in Section 3.1

2.2 Database Integration

The Web domain and the Life Science domain are two of the most active domains in integrating databases. Since these domains have many resources to handle (such as webpages or genomes) and since those resources are often stored separately for each project, there are strong demands for data integration and data exchange.

The Semantic Web [9] is a project initiative of the W3C to integrate and exchange Web resources. Web developers can use metadata to specify titles, publishers, meanings, and other semantic roles. The metadata is described in a Resource Description Framework (RDF) [6] or using the Web Ontology Language (OWL) [5]. By adding such metadata, applications handling RDF or OWL can interpret the meaning of Web resources, and they can also handle resources with similar meanings. For example, if two online banking websites have the same metadata, one application can use them equally well even though they may use different visual layouts or structures. Since the metadata is written in one format, it is not necessary to convert the data format, so the data exchange is relatively easy.

In the Life Science domain, integrating genome databases is an active area. YeastHub [13] is a project aiming to integrate many databases of yeast genomes. In the past, each yeast genome project has had its own database for storing its yeast genome data. Users can now search for yeast genome data in the YeastHub, and the results are tables or RDF that combines the data stored in the separate databases. However, since the data formats are usually unchanging and since the stored data is easy to convert, the data integration is relatively easy.

In contrast, Accessibility metadata does not have any fixed format. The formats vary from application to application, and can be tables, XML, scripts, etc. When integrating accessibility metadata, it is challenging to support the many formats.

3. ACCESSIBILITY COMMONS

We propose the *Accessibility Commons (AC)* infrastructure to integrate and share the metadata produced by the variety of research projects, assistive technologies, and individuals working in this area. The AC infrastructure consists of two key components: (i) a metadata repository with flexible metadata schema and (ii) a metadata conversion mechanism.

Metadata flexibility and metadata reusability are two of the key features of our metadata repository. Since existing metadata is not uniform and since the future form cannot be predicted, the schema of metadata database should be flexible enough to store most kinds of metadata. We have defined the schema of the metadata database so that most applications using metadata for Web accessibility can exploit the repository. The amount of available metadata and the number of people actually using metadata are the key metrics of success for such an infrastructure. The number of people benefitting from the infrastructure will increase with the number of applications using common metadata, because an existing metadata framework will free the developers from having to create their own. Having a well-established common metadata framework could also accelerate research projects. However, unexpected forms of metadata may appear in the future as new types of applications are invented. Therefore the proposed schema is not a final one. Our metadata schema is flexible enough to be extended as such metadata appears.

In addition, we provide a system for metadata conversion to accelerate the growth of metadata. By converting the metadata from each application's preferred format to a common format when possible, the amount of available metadata will greatly increase.

We discuss the AC metadata schema and the conversion system here.

3.1 AC Metadata Format

From our investigation of research projects and screen readers, we found that metadata consists of primary components:

- URI Addressing: A target URI specifying the documents for the metadata
- Element Addressing: A description of the HTML elements that the metadata applies to
- Condition: Conditions on the use of the metadata
- Semantics: What the metadata means

Table 1 shows the types of metadata used in each application.

Table 1: Comparison of metadata projects and screen readers

	URI Addressing	Element Addressing	Conditions	Main Semantics
aiBrowser	wild card	XPath	Dynamic	Heading, alternative text, transcoding
HearSay	domain matching, exact matching	XPath, URI	XPath	Labeling
WebInSight	-	URI, MD5	-	Alternative text
Site-wide Annotation	regular expression	XPath	XPath	Transcoding
Social Accessibility	wild card	XPath, URI	-	Heading, alternative text
JAWS	domain matching	relative path	-	Alternative text
AxsJAX	wild card	By program	By program	WAI ARIA
Accessmonkey	wild card	By program	By program	Alternative text, transcoding

The main challenge in defining a metadata format is making various types of metadata live together in the metadata repository. Over the years that external metadata has been studied, various kinds of metadata have been invented to cope with different of problems. A common metadata format needs to be flexible, interoperable, and extensible. We now review each type of the metadata in more detail.

3.1.1 URI Addressing

A URI is often used to target metadata. The flexibility and search efficiency of URI addressing is very important. Existing metadata adopts various forms of URI addressing to specify the pages that the metadata applies to. For example, Site-Wide Annotation uses regular expressions, aiBrowser uses a wild-card format, and JAWS does domain matching, specifically matching only the domain part of the URI. To be flexible for multiple forms of addressing, we represent the URI addressing as pairs of an addressing type (`URI_TYPE`) and a content identifier (`URI_PATTERN`). This can represent URI addressing without any loss of information.

The URI addressing is also used to filter the metadata when the metadata server returns query results. A client requests metadata from the server by sending information including a URI for the metadata the client wants. Then the server uses the URI sent by the client to query the AC database in the `URI_PATTERN` field. With this method, an application can use metadata just by querying for it, even if the application does not know the precise form of the URI addressing. However, this can cause a high load for the server because database indexing cannot be used. An index cannot be used for pattern matching, but it *can* be used for prefix matching. Indexes must be used for the URI matching to achieve high performance.

To use index-based searching, we extract part of the `URI_PATTERN` for which indexing is possible. In this case, we have chosen to extract the domain part of the URI, because wild cards or regular expressions in the domain part are rarely used, except when a subdomain is specified as a wild card or regular expression. The subdomain is often changed for several reasons, such as load balancing. We can extract `*.ibm.com` from `http://*.ibm.com/*`. However, prefix matching cannot be used when a wild card is used in the subdomain. An easy solution is to reverse the order and store `com.ibm.*` without any wild card. This allows us to use prefix matching, so that the reversed domain name can be indexed even if the subdomain changes. Since the domain can be shared by many kinds of metadata, it is stored in a different table.

A limitation of this method is that it requires the wild card or the regular expression to be used at the beginning of the domain. Other expressions, such as `www.*.com` and `www.ibm.*`, are prohibited. But these expressions rarely appear, so it does not matter in practical cases.

3.1.2 Element Addressing

Element addressing is used to specify elements of the target documents. The method of element addressing varies among applications. Examples of common element addressing techniques are: XPath, image/video URI, MD5 of image or video, text matching, and style matching (using CSS). Since each element addressing method has advantages and disadvantages, the adopted types of element addressing vary by application. For example, aiBrowser can understand only XPath, and JAWS does relative URI path matching. As with URI addressing, an addressing type (`ADDRESS_TYPE`) and a content address (`ADDRESS_DESC`) are used to represent element addresses.

Element addressing is often expressed as a combination of addressing techniques in some applications. For example, WebInSight uses a combination of an image URI and an image MD5 to specify an image element, because the image URI cannot detect that the image has changed and the image MD5 is not human readable. Therefore the metadata schema must be able to specify two or more forms of element addressing.

3.1.3 Conditions

Metadata should be able to specify conditions to check to verify when specific metadata is applicable. Conditions are important information for optimizing the range of metadata to where it is helpful. For example, A Web server occasionally returns different content even when the URI is unchanged. It may inject a new advertisements, or optimize its Web content for a client browser by using the `ACCEPT_LANGUAGE` header or the `USER_AGENT` header. In such cases, the conditions can improve the quality of the metadata. Site-wide Annotation uses conditions to choose the applicable metadata, checking the existence of the elements specified by the XPath.

It should be possible to specify multiple conditions and to use logical connectives such as AND.

Table 2: Metadata Semantics

Declarative Metadata	
Semantics	**Used Projects**
Alternative text	aiBrowser, WebInSight, …
Heading	aiBrowser, Social Accessibility, ….
Text insertion	aiBrowser, Social Accessibility, …
Transcoding	aiBrowser, Accessmonkey, …
Element grouping	HearSay
Element labeling	HearSay
Concepts	HearSay
Language	HearSay
Dynamic content	AxsJAX, JAWS, …
Procedural Metadata	
Semantics	**Used Projects**
Fennec†	aiBrowser
AxsJAX Script	AxsJAX
Accessmonkey Script	Accessmonkey
Recording Macros	HearSay

†the name of aiBrowser's transcoding engine

3.1.4 Semantics

Semantics describes the meaning of metadata. The semantics is expressed as both the type of semantics (`DATA_TYPE`) and its contents (`DATA_DESC`) similar to the other information of the metadata. However, the reusability cannot be ensured just by expressing the semantics in this form.

Some kinds of metadata are composed of mixtures of application-specific metadata and general-purpose metadata. This prevents us from reusing the general-purpose metadata, even though it should be reusable. For example, the metadata of aiBrowser is provided in one XML file, which includes headings, alternative texts, alternate contents, and other information.

When such combined metadata is instead stored separately, the reusability of the metadata can be improved because general-purpose metadata is easier to reuse in other applications. This will lead to overall accessibility improvements.

According to our investigation, the general-purpose metadata is mainly declarative. In other words, it only adds extra information to resources, and does not involve complex processing. In contrast, the application-specific metadata is often procedural. Table 2 shows a list of declarative metadata and procedural metadata. We must separate these types of metadata.

To store them separately, client cooperation is required, because the metadata server does not know about the structure of the application-specific metadata. The metadata server cannot distinguish between declarative metadata and procedural metadata. Therefore, clients should separate these kinds of metadata when submitting the metadata to the server. We will show an example of aiBrowser metadata in Section 5.1.

Table 3: Database Repository Schema

TABLE: METADATA		
NAME	**TYPE**	**EXAMPLE**
ID	INTEGER	3212
DOMAIN_NAME_ID	INTEGER	102
URI_TYPE	VARCHAR(8)	wildcard
URI_PATTERN	TEXT	http://www.ibm.com/*
DATA_TYPE	VARCHAR(8)	alttext
DATA_DESC	TEXT	IBM Server
ORIGINAL_URI	TEXT	http://www.ibm.com/some.html
AUTHOR_ID	INTEGER	12
TIMESTAMP	DATETIME	2008-05-01 (Thu) 13:30:20
AVAILABLE	BOOL	True
TABLE: ADDRESSES		
NAME	**TYPE**	**EXAMPLE**
ID	INTEGER	10413
METADATA_ID	INTEGER	3212
ADDRESS_TYPE	VARCHAR(8)	XPath
ADDRESS_DESC	TEXT	/HTML/BODY/DIV[3]/DIV[1]/P
TABLE: CONDITIONS		
NAME	**TYPE**	**EXAMPLE**
ID	INTEGER	242
METADATA_ID	INTEGER	3212
CONDITION_TYPE	VARCHAR(8)	XPath
CONDITION_DESC	TEXT	/HTML/BODY/DIV[2]/DIV[1]/P
TABLE: DOMAIN_NAMES		
NAME	**TYPE**	**EXAMPLE**
ID	INTEGER	102
DOMAIN	VARCHAR(255)	com.ibm.www

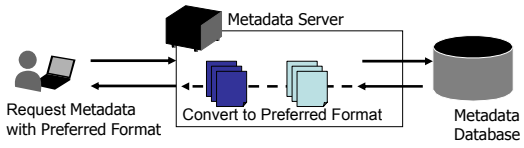

Figure 1: Metadata Conversion

3.2 AC Metadata Schema

Table 3 shows the metadata database schema for the Accessibility Commons infrastructure. As mentioned in the previous subsection, since multiple element addressing and logical conditions should be supported, they appear in another table.

The metadata may become obsolete when the webpage is updated and its structure is changed. In that case, we need to edit the metadata. To make editing easier, the original webpage URI, which is not a wild-card pattern or regular expression, is included in the schema. The `AVAILABLE` field can be used to disable specified metadata. The `AUTHOR_ID` field and a `TIMESTAMP` field are also added to the schema. `ID` is a primary key of each table.

3.3 Metadata Conversion

Each application can only handle the metadata which is known to it. For example, an application that cannot understand an XPath expression cannot exploit the metadata whose element addressing is in an XPath expression. This may decrease the amount of the metadata that can be used by each application. In order to cope with this problem, we provide server-side metadata conversion. Figure 1 shows an overview of the conversion system. Ideally, all of the AC clients should be able to use all of the the metadata without conversion. However, the server-side conversion has

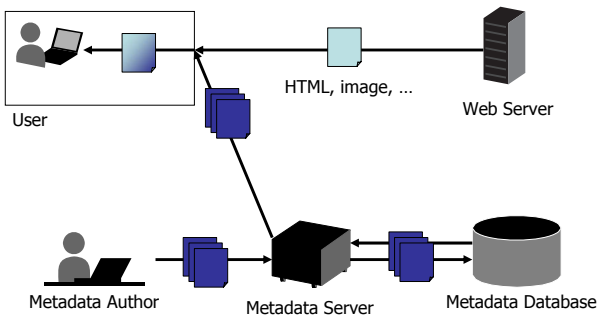

Figure 2: System Overview

advantages. When the server-side conversion is not provided, all of the clients must implement metadata conversion systems. This would be expensive for application developers, because the developer has to implement a conversion system that can handle all of the the types of metadata and the client would need to be updated when a new type of metadata appears. The initial costs and maintenance costs would encourage each application to focus only on its own metadata. To avoid this situation, the server-side conversion is important.

Here is an example of this conversion. When an application requires a regular expression format for URI addressing, the system converts a URI written in wild-card format, such as

`http://www.ibm.com/*`

to a regular expression format, such as

`http://www\.ibm\.com/.*`

In a similar way, if XPath is required in the element addressing, then the system converts an image URI to an XPath expression such as `//IMG[@src='...']`.

Of course, some metadata cannot be converted to some formats. For example, a wild-card format cannot be converted to a domain format in all cases and therefore cannot be used by applications that only understand domain matching. In this case, the system must ignore that metadata when it is requested by an incompatible client.

4. PROTOTYPE SYSTEM

We have implemented a prototype of the proposed Accessibility Commons infrastructure. The metadata server is implemented using Ruby on Rails [7] with a MySQL [8] database backend. The schema of the AC metadata was described in Section 0

Figure 2 shows a system overview. A metadata author is a person or an application that creates metadata to submit to the metadata server. A user is a person or an application that uses metadata from the server. The user can convert HTML documents to accessible documents or create annotations by using metadata taken from the metadata server.

When a client application submits metadata, the client uses JavaScript Object Notation (JSON) to represent the content of metadata. This contains all of the content of the metadata. The server system receives it and generates the metadata `ID`, `TIMESTAMP`, and `AUTHOR_ID` to store with it in the database. Then the server returns the metadata `ID` to the client.

When a client application retrieves metadata from the metadata server, it sends the URI of the metadata that it wants and the required types for that information. For example, the client can send a request such as the following:

```
{"uri"          : "http://www.ibm.com/",
 "uri_type"     : "wildcard",
 "address_type": "xpath"}
```

Then the server does URI matching, converts the matched metadata corresponding to the URI into the requested format for the client using the adaptive metadata conversion, and returns the response in JSON such as:

```
 [{"id"           : 1024,
   "uri_type"     : "wildcard",
   "uri_pattern"  : "http://www.ibm.com/*",
   "data_type"    : "alttext",
   "data_desc"    : "IBM Server",
   ...}, …]
```

The client then uses the requested metadata. If the adaptive metadata conversion system cannot convert all of the metadata, then the client should skip the metadata that cannot be handled.

The prototype system now works as a backend of the Social Accessibility Project [19]. Currently our server is in an alpha stage. We plan to release a public beta version late this year.

5. AC Use Examples and Scenarios

In this section, we present several use cases and scenarios for the Accessibility Commons infrastructure. With aiBrowser, we show how to extract reusable metadata for other applications from the aiBrowser-specific metadata with the client cooperation. We also discuss how the HearSay audio browser prototype and SADIe uses the AC schema to store their metadata. We believe that these scenarios show the flexibility and reusability of the AC metadata and the AC infrastructure.

5.1 Case: aiBrowser

The aiBrowser can create accessible alternate content by transcoding dynamic content based on the aiBrowser metadata. To the best of our knowledge, this feature is implemented only in the aiBrowser. The aiBrowser metadata is a mixture of application-specific metadata and general-purpose metadata, so storing them separately will increase the reusability of the metadata.

The following XML is a part of the aiBrowser metadata.

```
<node loc:path="id('heading')/DIV[2]">
  <h1 />
  <node loc:path="./IMG[1]">
    <altText>Mt. Fuji</altText>
  </node>
  <node loc:path="./IMG[2]">
    <altText><ref  loc:path="./P[2]" /></altText>
  </node>
</node>
```

The aiBrowser's metadata specifies the structure of alternate content as an XML structure. It also shows that an element represented in the XPath `id('heading')/DIV[2]` should be a heading, the first image of that element should have the alternate text "Mt. Fuji", and that the second image of that element should have alternate text whose text is taken from `./P[2]`.

In this example, the alternative text referring to another element is in a notation specific to aiBrowser, and it is difficult to share it with other applications. However, specifying a heading tag and adding static alternative text is an example of general-purpose metadata, and they can be readily shared with other applications.

The general-purpose metadata part is the nodes whose element addressing can be expressed in XPath and for which the description of the metadata does not refer to any other content. When converting aiBrowser metadata to the common form, we extract the general-purpose metadata first and submit it to the metadata server. Then, a metadata id is returned from the server, we replace each element with `<external refid="id" />`. In this case, the following metadata was created from the original metadata. (Here, the extracted metadata ids are 20 and 21)

```
<external refid="20">
  <external refid="21" />
  <node loc:path="./IMG[2]"><altText>
    <ref loc:path="./P[2]" />
  </altText></node>
</external>
```

This metadata is submitted to the server with the `DATA_TYPE` "fennec" and the `DATA_DESC` in XML. When aiBrowser converts AC metadata to aiBrowser metadata, it replaces the `external` elements with the common metadata. When aiBrowser uses other applications' metadata, it creates more aiBrowser metadata. For example, metadata for image alternative text can be used like this:

```
<attach loc:path=".">
<node loc:path="//IMG[@src='http://.../foo.png']">
  <altText>Flower</altText>
</node>
</attach>
```

This means the alternative text for all `IMG` elements with URI of `http://www.ibm.com/foo.png` is ``Flower''

5.2 Case: HearSay

The HearSay research team (Stony Brook University, USA) is working on automated techniques for content analysis, aiming to help blind users find relevant information in webpages more quickly and to make non-visual Web browsing more usable and efficient. The modular architecture of the HearSay Web browser allows plugging in different algorithms, each of which produces annotations or labels for Web content. The browser then uses the labels while generating its VoiceXML dialogs and provides a usable interface to navigate between the labels [16].

Since the automated techniques cannot guarantee 100% accuracy, the HearSay team developed a prototype module that allows users to override the results of automatic labeling with manual annotations and to benefit from social networking by sharing labels with other users. Initially, the HearSay team had independently designed a database schema for storing and sharing the labels. However, following a joint effort to create a common infrastructure, HearSay has migrated to the Accessibility Commons (AC) infrastructure and will continue using AC schema for storing its metadata.

HearSay currently uses labels to mark the beginning of the main content [18], the source language, the concepts required for Web transactions [17], the dynamic content [16], and free text labels. The HearSay team is planning to add support for headings and alternative text, and is also looking for ways to extend the AC metadata to support ARIA content and recorded macros.

5.3 Case: SADIe

While other use cases have looked at data consumption and individual metadata deposits, SADIe can help with large scale site-wide metadata creation. By using CSS and the Web Ontology Language (OWL) technology already present within the tool, we can automatically create site-wide metadata for end tool consumption. This is also advantageous in that the metadata for sites that change their structures can be quickly regenerated by reapplying SADIe. More importantly, extra information not already present within the website semantics can be machine readable using bridges between SADIe and other tools, such as HearSay, provided by the metadata repositories. The metadata repository enables disjoint technologies to work in concert for outcomes not possible individually.

6. DISCUSSION AND FUTURE WORK

We discuss inevitable spam attacks and conflicts in the metadata, and the performance of the database here.

6.1 Conflicts and Broken Metadata

The accumulated metadata may have conflicts. For example, two different alternative texts may be supplied for the same images. Also, the repository may contain broken metadata, such as incorrect metadata and meaningless metadata, because the webpages that the metadata is applied to changes and the element addressing of the metadata does not work. This will make users confused. In such cases, what should we do? Currently the AC infrastructure does not detect any conflicts or broken metadata. The system returns all of the corresponding metadata when objects asked for it. Which metadata should be used is left to the client. As for broken metadata, the system expects that all the metadata is maintained by someone. When the amount of metadata user is large enough, such broken metadata will be reported in a short time, and it will be fixed, so we don't think this is a major problem. For conflicts, the approach that all of the related metadata is returned has pros and cons. Since a client that does not consider conflicts may apply conflicting metadata randomly, the user may become confused when using such a client. However, a smart client may be able to choose the better metadata by using context analysis or some other approaches. We would like to leave such possibilities open. In addition, since this server provides the metadata author using the `AUTHOR_ID` field, a client can determine who created the metadata. If a reputation system is available, then the client can choose the metadata supplied by the most reliable author. Alternatively, if a client knows which author is an automatic analysis engine, the client can give that metadata lower preference relative to human authored metadata. Also, the client can choose the latest metadata, which may fit the current page. Of course, these kinds of information can be provided with this infrastructure.

6.2 Spam Metadata

The possibility of spam attacks, submitting lots of meaningless or broken metadata, exists in this infrastructure. Though we definitely have to detect and reject them, it is impossible to cover all of them. However, it is possible to reduce the damage by introducing some protection mechanisms, such as a limited number of queries per second. Currently this is left as future work.

6.3 Performance of Database

Good performance of the database is crucial for the infrastructure. It is expected that millions of metadata records will accumulate in the common repository, and this growth may negatively affect the performance of the repository. Users dislike unresponsive systems. As described in Section 3.1.1, we chose to use domain names as an index for the database. Therefore the performance may depend on how much metadata exists in a domain. For the Site-wide Annotation [20], 245 annotations files were used to transcode

USAToday.com, and the processing time was reasonable. Since the granularity of the AC metadata is finer than that of Site-wide Annotation, a larger amount of metadata may be recorded in each domain. However, considering the improvements of hardware and networks, we believe that our infrastructure will be able to process user queries with acceptable response times. In addition, if the client caches metadata, the processing time can be drastically reduced, because the client can query the difference between cached metadata and the latest metadata. Also, since metadata queries can be processed in parallel, it is easy to enhance the infrastructure as required.

7. CONCLUSION

In this paper, we have reviewed existing research projects concerning metadata and non-visual Web browsing, and we have proposed and prototyped a common metadata infrastructure, called *Accessibility Commons (AC)*, for the metadata integration and sharing. Thanks to this infrastructure, Web accessibility can be improved by merging the contributions of the existing but disconnected research projects. This infrastructure consists of two components: (i) a common metadata repository with a flexible metadata schema and (ii) a metadata conversion system. The repository is designed to accumulate most types of metadata produced by the various research projects, assistive technologies, and individuals. The conversion system can convert the metadata to each application's preferred format so that the cost of developing AC clients can be reduced.

We hope that the AC infrastructure can accelerate accessibility research projects and the development of accessibility applications to improve the real usability. Our goal is to standardize the AC metadata to effectively deploy the infrastructure. So that existing accessibility projects and future projects will easily adapt it. Our next technical challenge is to broaden the applicability of the AC metadata. We will strive to support dynamic contents, such as AJAX and Flash to leverage the value of the AC metadata and deliver useful and accessible information to real users.

8. ACKNOWLEDGMENTS

We thank Daisuke Sato and Takashi Ito for their contributions to the implementation of the prototype systems. We also thank Kentaro Fukuda and Naohiko Ohkohchi for testing our system. Finally, we would like to thank our anonymous reviewers for their comments and suggestions.

9. REFERENCES

[1] JAWS for Windows Screen Reading Software http://www.freedomscientific.com/products/fs/jaws-product-page.asp

[2] Fire Vox: A Screen Reading Extension for Firefox http://firevox.clcworld.net/.

[3] Adobe Flash. http://www.adobe.com/products/flash/

[4] AxsJAX. http://code.google.com/p/google-axsjax/

[5] OWL. http://www.w3.org/TR/owl-features/

[6] RDF. http://www.w3.org/RDF/

[7] Ruby on Rails. http://www.rubyonrails.org/

[8] MySQL. http://www.mysql.com/

[9] W3C Semantic Web Activity. http://www.w3.org/2001/sw/

[10] WAI ARIA. http://www.w3.org/TR/wai-aria/

[11] J. P. Bigham, R. S. Kaminsky, R. E. Ladner, O. M. Danielsson, and G. L. Hempton. WebInSight:: making web images accessible. In *ASSETS '06: Proceedings of the 8th international ACM SIGACCESS conference on Computers and accessibility*, 181-188. 2006.

[12] J. P. Bigham and R. E. Ladner. Accessmonkey: a collaborative scripting framework for web users and developers. In *W4A '07: Proceedings of the 2007 international Cross-Disciplinary Conference on Web Accessibility*, vol. 225, 25-34. 2007.

[13] K. H. Cheung, K. Y. Yip, A. Smith, R. deKnikker, A. Masiar, and M. Gerstein. YeastHub: a semantic web use case for integrating data in the life sciences domain. In *Bioinformatics 21, 1 (Jan. 2005)*, 85-96. 2005.

[14] H. Miyashita, D. Sato, H. Takagi, and C. Asakawa. aiBrowser for multimedia: introducing multimedia content accessibility for visually impaired users. In *ASSETS '07: Proceedings of the 9th international ACM SIGACCESS conference on Computers and accessibility*, 91-98. 2007.

[15] I. V. Ramakrishnan, A. Stent, and G. Yang. Hearsay: enabling audio browsing on hypertext content. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, 80-89. 2004.

[16] Y. Borodin, et. al. Towards One World Web with HearSay3. In *W4A'08: Proceedings of the 2008 international Cross-Disciplinary Conference on Web Accessibility*, 128-129. 2008.

[17] J. U. Mahmud, Y. Borodin, and I. V. Ramakrishnan. Assistive Browser for Conducting Web Transactions. In *IUI '08: Proceedings of the International Conference on. Intelligent User Interface*. 2008.

[18] J. U. Mahmud, Y. Borodin, and I. V. Ramakrishnan. Csurf: a context-driven non-visual web-browser. In *WWW '07: Proceedings of the 16th International Conference on World Wide Web*, 31-40. 2007.

[19] H. Takagi, S. Kawanaka, M. Kobayashi, T. Itoh, and C. Asakawa. Social Accessibility: Achieving Accessibility through Collaborative Metadata Authoring. In *ASSETS '08: Proceedings of the 10th International Conference on Computers and Accessibility*. 2008.

[20] H. Takagi, C. Asakawa, K. Fukuda, and J. Maeda. Site-wide annotation: reconstructing existing pages to be accessible. In *ASSETS '02: Proceedings of the fifth international ACM conference on Assistive technologies,* 81-88. 2002.

[21] S. Harper and S. Bechhofer. SADIe: Structural semantics for accessibility and device independence. In *TOCHI '07: ACM Transactions on Computer-Human Interaction*, 10. 2007.

[22] E. J. Chikofsky and J. H. C. II. Reverse engineering and design recovery: A taxonomy. In *IEEE Software*, 13-17. 1990.