



Web Architecture and Technology

A Standard for Indian Language Encoding and Fonts with Case-sensitivity

Alka S Irani[#], Minoo Dosabhai[#], Zia Saquib[#], P D Bhatnagar^{*}

[#]Centre for Development of Advanced Computing
Gulmohar Cross Road, Juhu, Mumbai, 400 049, India

¹alka@cdacmumbai.in

³saquib@cdacmumbai.in

^{*}Consortium of Scientists for Sustainable Development
4649/21, Darya Ganj, New Delhi-02, India

⁴scientistpd@gmail.com

Abstract— For translating from one language to another the distinction between proper names and language words is an important one. Language words are ‘translated’ while proper names are ‘transliterated’. By providing two cases (upper case and lower case) and by starting the proper names with upper cases, roman alphabet based languages like English provide a syntactic device to distinguish language words and proper names (ambiguities still exist because the first letter of the first word of the sentence is also written in uppercase).

If Indian languages on computers could follow some convention for differentiating proper names and language words, it will immensely help translations across various languages. For this we can have the concept of uppercase or capital Devanaagarii letters that closely resembles the conventions followed by

natural language ‘as it is’ presents many problems. The problems can be characterized as problems due to ambiguity, problems due to fuzziness of symbols, problems with context sensitivity and problems with idiosyncrasies of the languages (too many usages).

If we process Indian language text on computer in absence of capitalization, a major issue for the computer programs will be how to differentiate between language words and identifiers (proper names).

A. Use of Capitalisation in a Roman script based languages like English

In English, for each letter of the alphabet (a-z), its capital form (A-Z) exists and it has become the part of the basic alphabet. Thus everyone who learns the script simultaneously learns the Upper case form

in other languages (e.g., French), namely to use sentence-style capitalization in titles and headlines, where capitalization follows the same rules that apply for sentences.

We strongly recommend sentence case for titles and headings as well.

F. Computers and Case sensitivity

Some sentence cases are not used in standard English, but are common in computer programming, as well as in product branding and in other specialised fields:

1. **Upper CamelCase:** First letter of each word is capitalized, spaces and punctuation removed.
2. **Lower camel case** describes a variation, as in “

Table I. The complete scheme for Case Sensitive Devanagari

Devanagari upper & lower case	Rupantar keyboard [10]	Diacritic equivalent encoding
□ □ □ □ □	a aa i ii u uu	a ā ī ū ū
□ □ □ □ □ □ □ □	e ai o au .m :h :e :o	e ai o au □ □ ē ō
□ □ □ □ □ □ □ □	k kh g gh .gh	k kh g gh □ □
□ □ □ □ □ □ □ □	ch chh j jh .jh	ch chh j jh ñ
□ □ □ □ □ □ □ □	.t .th .d .dh .n	□ □ h □ □ h □
□ □ □ □ □ □ □ □	t th d dh n	t th d dh n
□ □ □ □ □ □ □ □	p ph b bh m	p ph b bh m
□ □ □ □ □ □ □ □	y r l v sh	y r l v sh
□ □ □ □ □ □ □ □	.s s h .l .xh .yh	□ s h □ .xh .yh
□ □ □ □ □ □ □ □ □ □	ka kaa ki kii ku kuu	ka kā ki kī ku kū
□ □ □ □ □ □ □ □ □ □	ke kai ko kau ka.m ka:h	ke kai ko kau k□ k□

I. Encoding and case folding

1. The uppercode distinction is required not be only while displaying but it should also be preserved in the file. Thus character encoding should incorporate the feature 'lower case and upper case'.
2. At times when we do not want case sensitivity the uppercase and corresponding lowercase character should be treated as equivalent. i.e. Case-folding should be possible.

This can be handled by having a suitable encoding scheme for Indian languages. In fact we feel, if we standardise a convention for encoding all Indian languages in diacritic form, that itself can form a suitable encoding scheme for Indian languages on computers.

IV. SUMMARY AND CONCLUSION

Though natural languages have evolved independently, there are lot of similarities and some differences. For natural languages on computers we need not build the medium from scratch but must do incremental changes for essential functionalities. Standardising a convention makes the medium a lot more acceptable. For Indic languages we do not follow the convention of having first word of the sentence capitalised. We also need not follow the convention for title case. In fact, we can have the standard basically to distinguish two kinds of entities, those which get translated and those which get transliterated when changed from one language to another. We believe this convention will closely correspond to the semantic partitions that exist in our mind.

ACKNOWLEDGEMENTS

To all those with whom we were associated over years especially from the institutes C-DAC Mumbai (erstwhile NCST), C-DAC, TIFR, TDIL (Govt of India), Bharati Samskrit Vidya Niketanam, Consortium of Indian Scientists for Sustainable Development, ICSW, Govt of Maharashtra, MTNL, Air India and BITS Pilani (Rajasthan) where the first author had done her Ph.D. and of course the families, friends and the supreme power.

REFERENCES

- [1] The Unicode Standard Version 3.0 and 5.2, Unicode Consortium.
- [2] Eugene Ehrlich "Theory and problems of Punctuation, Capitalization and Spelling 2nd Ed", Schaum's Outline Series, McGraw-Hill Inc,
- [3] The Chicago manual of Style, 13th Edition, The University of Chicago, revised and expanded, Pretice-Hall of India, 1989.
- [4] The Oxford Universal Dictionary on Historical Advanced Proportional Principles (reprinted 1952)
- [5] Oxford Manual of Style" (2002),
- [6] Stuart Russell and Peter Norvig, *Artificial Intelligence – A modern approach* : Prentice Hall
- [7] Steven L Small, G W Cottrell and M K Tanenhaus Ed., *Lexical Ambiguity Resolution: Perspectives from Psycholinguistics, Neuropsychology, and Artificial Intelligence* Morgan Kaufmann Publishers, Inc., California, 1988
- [8] Alka S Irani, "A unified model for concept structuring", Ph.D. thesis, BITS Pilani, Rajasthan, India, 1996
- [9] (2010) The Wikipedia website, [Online], Available: <http://www.wikipedia.org>
- [10] Alka S Irani, "Rupantar – Transliteration and an Intermediate coding scheme for Indian languages", white paper C-DAC Mumbai, 2004

Dynamic Slicing of Service-Oriented Architecture

Kaushik Rana, Durga Prasad Mohapatra

Dept. of CSE, NIT, Rourkela, India.
U.V. Patel College of Engineering, Kherva, India.
kaushik.rana@ganpatuniversity.ac.in
durga@nitrkl.ac.in,

Abstract— Service-Oriented Architecture (SOA) is a novel design methodology to revolutionize every major computing platform that is being supported and adopted world wide. Testing of services is a complex activity which carries major concerns for both service providers and consumers. This paper introduces the dynamic slicing technique for testing of service, after assuming the availability of service code. In this context, the WS-AtomicTransaction framework is utilized and extended to give support in terms of coordination type, which we called WS-Testing. This framework introduces new dimension to coordination context by including testing related data. Further, it requires the testing task to be decomposed into various subtasks like gathering test context information, preparing services, and testing services. The novelty of our approach lies in the way decomposition is carried out on testing activity and reusability of existing frameworks and protocols to support it and the extension to the coordination context information.

I. INTRODUCTION

The sky-rocket adaption growth of Service-Oriented Architecture (SOA) demands novel approach for testing. Since, services are exposed over the internet; their failures cause the significant loss of business therefore, appropriate guarantees need to be provided that they will behave properly.

A few frameworks have been reported for testing service. Hong Zhu[1] proposes a service oriented framework for testing web service, for n service it requires n^2 additional services. W.T.Tsai et al[2] propose an object-oriented framework for testing service based on test master and test engine. Cesare Bartolini et al.[3] propose approach for testing of service based on coverage information, which is not sufficient to provide test information at runtime.

II. WS-TESTING FRAMEWORK

Testing of SOA is mainly challenged by loose coupling, an inherent relationship achieving awareness through the use of service description and high dynamism.

This paper proposes an extension to the WS-AtomicTransaction framework, WS-Testing to accommodate testing of service using dynamic slicing technique. When WS-Testing is used, the atomic transaction coordinator can be referred as dynamic slicer. The dynamic slicer as shown in Figure 1 plays a key role in managing, preserving, and distributing the testing context information among the participants. The testing context information includes: testing activity identifier, expiration value, token, error message etc.

2.1 The Activation and Registration Process

The testing activity with the use of dynamic slicing technique is being decomposed into following three major tasks.

1. Gather testing context information.
2. Prepare service
3. Test service

Each of the above major tasks can be decomposed into one or more steps as following:

1. Gather testing context information consists of:
 - 1.1 Lexical analysis
 - 1.2 Error handler
 - 1.3 Parsing
2. Prepare service consists of:
 - 2.1 Service code instrumentation
3. Test service consists of:
 - 3.1 Service invocation
 - 3.2 Generate dynamic slices w.r.t slicing criterion.

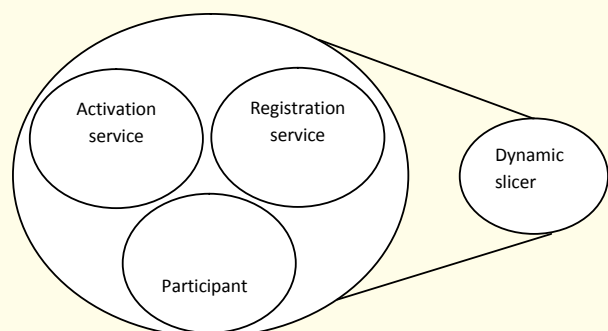


Fig. 1. The dynamic slicer

The dynamic slicer control the composition of the following two other services that each plays a specific part in management of testing context information. The dynamic slicer also assigns responsibility to each composition member.

- (1) Activation service—responsible for carry out the Gathering of testing context information.
- (2) Registration service—allows participating services to use testing contexts information received from activation service to register for an atomic transaction protocols (completion, volatile 2PC, durable 2PC).

The service that has completed registration is considered as participant.

II. CONCLUSIONS

The proposed WS-Testing framework reuses existing WS-AtomicTransaction framework, an inherent characteristic of service based systems. The context information is more capable to provide test related information. The implementation of WS-Testing framework is in first stage.

REFERENCES

- [1] Hong Zhu. A Framework for Service-Oriented Testing of Web Services, In the Proceedings of the 30th Annual International Computer Software and Applications Conference (COMPSAC'06), 2006.
- [2] W.T.Tsai, Ray Paul, Weiwei Song, Zhibin Cao. Coyote: An XML-Based Framework for Web Services Testing, In the Proceedings of the 7th IEEE International Symposium on High Assurance Systems Engineering (HASE'02).
- [3] Cesare Bartolini, Antonia Bertolino, Eda Marchetti, Introducing Service-oriented Coverage Testing.

XML NAMESPACES FOR DOCUMENT IMAGE PROCESSING

Sanghamitra Mohanty
sanghaml@rediffmail.com

Himadri Nandini Das Bebartta
himadri_nandini@yahoo.co.in
Dept. of Computer Science & Application,
Utkal University, Bhubaneswar,
Orissa, India- 751004

Tarun Kumar Behera
tarun06@gmail.com

Abstract- XML namespace reduces the risk of name collisions by taking advantage of existing systems for allocating globally scoped names: the URI system. When using XML namespaces, each local name in an XML vocabulary is paired with a URI which is known as the namespace URI to distinguish the local name from local names in other vocabularies. Another benefit of using URIs to build XML namespaces is that the namespace URI can be used to identify an information resource that contains useful information, machine-usable and/or human-usable, about terms in the namespace. Using the advantage of namespaces in XML we have tried to make use of it in our related research field i.e. document image processing in Oriya OCR. For the recognition of a printed document, the steps that are carried out are binarization, skew detection and correction, segmentation, feature extraction and recognition. For carrying out these phases we need to instantiate the schema parser first to read the XML file. The XML file consists of different XML elements constituting of different phases of the document to be processed. The syntactic correctness of the schema is then validated i.e. it checks if schema elements obey the structure in terms of allowed attributes and children of schema elements. In the next phase the schema elements are processed semantically. All the elements within the XML schema tag are analyzed. Parser picks every element in the XMLized document, searches its corresponding schema element and validates it. There we check the attributes, contents and verify the child elements for which we use the function, verify against schema. Implementation of these procedures has been performed in Oriya OCR.

I. INTRODUCTION

For the Internet-based data exchange solutions, Extensible Markup Language (XML) is rapidly becoming the technology of choice. Mainly it has a wide application in the field of information technology (IT). A key aspect in the development and deployment of XML is the use of namespaces. According to the World Wide Web Consortium (W3C), the purpose of XML namespaces is to provide a simple method for qualifying element and attribute names used in

Extensible Markup Language documents by associating them with namespaces identified by URI references [1]. Namespace associations allow XML implementers to use diverse XML vocabularies without fear of name collision resulting in invalid XML. To ensure consistency, organizations should decide on a namespace strategy and a naming convention when establishing organizational namespaces. The federal government is actively engaged in developing and deploying XML [2]. James Clark has written a document "XML Namespaces" which tries to explain the mechanism specified by the W3C XML Namespaces Recommendation. It explains things in a somewhat different way which [Clark] hopes at least some people may find less confusing than the explanation in the Recommendation [3]. "XML Namespaces by Example", by Tim Bray has shown the arrival of a new W3C Recommendation, Namespaces in XML. 'Recommendation' is the final step in the W3C process [4]. XML based representation is better suited for multimedia and document analysis tasks [5]. For printed document analysis, XML has been used earlier for tasks such as representation of annotated data sets [6,7]. Semantic web technologies [8] also use XML as their underlying data representation and exchange format. Scheme based upon XML based labeling for managing a large multilingual OCR project has also been done by G. Harit et. al [9]. Namespaces are a simple and straightforward way to distinguish names used in XML documents, no matter where they come from. The best way to understand namespaces, as with many other things on the Web, is by example.

Namespaces are a mechanism for managing names in a distributed way that greatly reduces the likelihood that two independent parties will create the same name for different purposes. The terms in a namespace are two-part identifiers consisting of a namespace name (a URI) and a local name as defined in (W3C XML Namespaces). Using a URI leverages the well-understood URI allocation mechanisms. Namespaces originally designed to provide names for XML elements and attributes have been adopted much more broadly by the web community. They are now used not simply for elements and attributes but for function names, tokens, and identifiers for ever more purposes. The xml namespace demonstrates that some namespaces

benefit from a policy that allows additional names to be defined in them over time.

II. NAMESPACES AND XML SCHEMA

The concept of XML namespaces is defined in the W3C XML namespaces technical specification [1]. Following these cocepta we have tried to implement the specifications in our Oriya OCR. XML namespace features are available in the W3C XML Schema (XSD) and in document-type definitions (DTDs). Because most new XML development work is being done using schemas and the Draft Federal XML Developer's Guide recommends using XSD, our discussions focus on the use of namespaces in XSDflO]. To help the reader understand the concepts involved and the reasons for our subsequent recommendations, the following subsections explain the concepts of namespace declaration, target namespaces.

XML namespace is a collection of XML elements and attributes identified by an Internationalized Resource Identifier also known as IRI. This collection is often referred to as an XML vocabulary.

One of the primary motivations for defining an XML namespace is to avoid naming conflicts when using and re-using multiple vocabularies. The XML Schema is used to create a vocabulary for an XML instance, and uses namespaces heavily. Thus, having a knowledge on namespace requires understanding of XML Schema and instance validation overall.

Though namespace is basically declared in the root element of a Schema using a namespace identifier, it is not mandatory to declare namespaces only at the root element always; rather it could be declared at any element in the XML document. This namespace identifier must be a URI reference that conforms to the Internet Engineering Task Force (IETF) Request For Comments (RFC) 2396, Uniform Resource Identifiers: Generic Syntax [11]. Schema constructs are associated with a namespace identifier through a user-defined namespace prefix, making the constructs "namespace qualified."

Namespaces are declared as an attribute of an element. The scope of a declared namespace begins at the element where it is declared and applies to the entire content of that element, unless overridden by another namespace declaration with the same prefix name—where, the content of an element is the content between the <opening-tag> and </closing-tag> of that element. A namespace in relation to our work is declared as follows:

```
<PAGE ImageURI="/root/Desktop/1.tif" ImageType="tif" ImageWidthpx="2000" ImageHeightpx="3000" >
```

The above example is read as binding namespace prefix ImageURI with the namespace "/root/Desktop/1.tif". Thus to use a namespace, we first bind it with a prefix and then use that prefix wherever required.

When a namespace is defined for an element, all child elements with the same prefix are associated with the same namespace. Namespaces can be declared in the elements where they are used or in the XML root element and we have made used of namespace only in the root element only.

```
<?xml version="1.0" encoding="UTF-8"?>
<DOCUMENT>
  <PAGE ImageURI="Output/ScreenDetection/0205_Paladei_img_600_Smooth_Page_0002.tif" ImageFormat="tif" ImageWidth="2788" ImageHeight="4568">
```

Fig. 1. Characters Observed For Classification

We can see from the above Fig. 1 that the URI is taken as ImageURI and the namespace associated with it is the output from a previously processed module. It is a convention to use XSD or XS as a prefix for the XML Schema namespace. Using meaningful namespace prefixes add clarity to the XML document. The prefixes are used only as a placeholder and must be expanded by the namespace-aware XML parser to use the actual namespace bound to the prefix. Although a namespace usually looks like a URL, that doesn't mean that one must be connected to the Internet to actually declare and use namespaces. In the Internet space URLs are unique—hence we would usually choose to use URLs to uniquely identify namespaces.

Like any other XML document, XML Schema is built with elements and attributes. This "building material" must come from the namespace http://www.w3.org/2001/XMLSchema, which is a declared and reserved namespace that contains elements and attributes as defined in W3C XML Shema Structures Specification and W3C XML Schema Datatypes Specifications.

We should not add elements or attributes to this namespace. Using these building blocks we can create new elements and attributes as required and enforce the required constraints on these elements and attributes and keep them in some namespace as we can see from Fig. 2 XML Schema calls this particular namespace as the {target namespace}, or the namespace where the newly created elements and attributes will reside.

During validation, the Validator verifies that the elements/attributes used in the instance exist in the

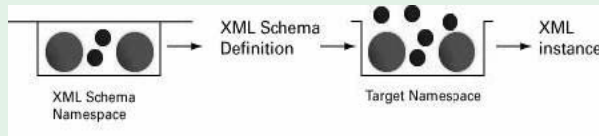


Fig. 2. Elements and attributes in XML Schema namespace are used to write an XML Schema document, which generates elements and attributes as defined by user and puts them in {target namespace}. This {target namespace} is then used to validate the XML instance.

declared namespace, and also checks for any other constraint on their structure and datatype.

III. PHASES PERFORMED FOR OCR

The major architectural components of an Optical Character Recognition system would be: Image acquisition, Image pre-processing/enhancement, Script-Independent processing and Script-dependent schemes. XML has been used as an architecture specification language for each of these above modules and sub-modules. Each of the modules get a XML file as input and parsing it they extract the relevant information required for their functioning. During the Image Acquisition, the input to our Oriya OCR is a raw image either scanned or taken from the corpus. The information related to the image that is needed by the OCR for choosing proper set of pre-processing routines are stored in the XML file. Mode (single or multi page), Image Location, Image Name, Image format, Image dimensions, Scanning Resolution (200, 300 and 600 dpi), Pixel info type (color, grayscale and binary) and Script Identification is a list of properties stored. Fig. 3 gives the XML storage schema. Enhancement of the quality is required for best result because while performing the Pre-processing techniques the input page may not be fit for applications involving character recognition. Major pre-processing techniques used are noise removal, skew correction, Binarization. Corresponding to each technique we can have XML tags with their algorithm specific attributes. The input to these pre-processing routines is the XML file which contains all the image related information. The routines parse the XML file and pick the component separated from the image like text, picture or graphics. image from the location specified in the XML ImageURI tag. Fig. 5 shows the sample xml schema used for separating the text XML Schema for the preprocessing modules is given in Figure and picture block below.

Here in Fig. 4 we can see the XML tags that is given for carrying out the skew binarization and skew detection module. In the xml tag, the specification is given that which algorithm we want to use for binarization and skew correction. Accordingly it runs



Fig. 3. XML Storage Schema



Fig. 4. XML Tags given for Binarization and Skew Correction

it by reading it from XML tags specified.

Script independent routine includes the processes such as Text non-text separation, Text extraction, Text graphics separation. These processes help in identifying picture/figure blocks, graphics and text blocks. It stores the bounding box information for each component in the XML tags (topLx, topLy, bottomRx and bottomRy) [9]. The routines also create tag related to type of

The text blocks are further segmented into lines and words using script dependent techniques. Storing all intermediate information and final results in XML helps in saving memory as well as for verifying the performance of recognition algorithms

IV. OUTPUT SPECIFICATION

The final output received from the OCR system is encoded using XML tags. The output XML file has the document layout structure, segmented regions of interest and their categorization (text, graphics, pictures) and the UNICODE stream for the words for the text extracted.


```
<?xml version="1.0" encoding="UTF-8"?>
<DOCUMENT>
  <PAGE ImageURI="0205_Patadei_img_600_Smooth_Page_0002.tif" ImageFormat="tif" ImageWidth="1381" ImageHeight="2275">
    <Binarization Threshold="150" AlgorithmName="Nishize">
      <InputImageURI>
        0205_Patadei_img_600_Smooth_Page_0002.tif
      </InputImageURI>
      <OutputImageURI>
        0205_Patadei_img_600_Smooth_Page_0002.tif
      </OutputImageURI>
    </Binarization>
    <Skew SkewAngle="0.000000" AlgorithmName="Nishize">
      <InputImageURI>
        0205_Patadei_img_600_Smooth_Page_0002.tif
      </InputImageURI>
      <OutputImageURI>
        0205_Patadei_img_600_Smooth_Page_0002.tif
      </OutputImageURI>
    </Skew>
    <TextBlock TotalNumber="1">
      <InputImageURI>
        0205_Patadei_img_600_Smooth_Page_0002.tif
      </InputImageURI>
      <OutputImageURI>
        0205_Patadei_img_600_Smooth_Page_0002.tif
      </OutputImageURI>
      <BLOCK Number="1">
        <topLx>
          1
        </topLx>
        <topLy>
          1
        </topLy>
        <bottomLx>
          2787
        </bottomLx>
        <bottomLy>
          8565
        </bottomLy>
      </BLOCK>
    </TextBlock>
  </PAGE>
</DOCUMENT>
```

Fig. 5. Sample XML Schema for Separating Text and Picture Block

The information content of the final XML output of the document-analysis-cum-OCR module is shown in Fig. 6 Firstly we consider the complete document image. Then after segmentation the entities involved are text regions, picture regions and graphics or tables. We consider only the text from it. Then segment those texts to words. And finally the OCR output unicode stream of symbols for the corresponding word.

```
<?xml version="1.0" encoding="UTF-8"?>
<DOCUMENT>
  <PAGE ImageURI="Output/SkewDetection/0205_Patadei_img_600_Smooth_Page_0002.tif" ImageFormat="tif" ImageWidth="2788" ImageHeight="4346">
    <TextBlock TotalNumber="1">
      <InputImageURI>
        Output/SkewDetection/0205_Patadei_img_600_Smooth_Page_0002.tif
      </InputImageURI>
      <OutputImageURI>
        Output/SkewDetection/0205_Patadei_img_600_Smooth_Page_0002.tif
      </OutputImageURI>
      <BLOCK Number="1">
        <topLx>
          0
        </topLx>
        <topLy>
          0
        </topLy>
        <bottomLx>
          2787
        </bottomLx>
        <bottomLy>
          4565
        </bottomLy>
        <Unicode FileURI="Output/Oriya/0205_Patadei_img_600_Smooth_Page_0002_1.txt">
          ...
        </Unicode>
      </BLOCK>
    </TextBlock>
  </PAGE>
</DOCUMENT>
```

Fig. 6. The Final Output Dumped after processing

In Fig. 5 we can see the output XML that we have obtained which displays the four coordinates that has been selected for processing.

V. CONCLUSION

In this paper we have provided a brief description on the techniques that can be performed to make use of XML Namespaces in Oriya OCR. By the help of XML Namespace we have performed all the phases of our Oriya OCR perfectly. The attributes, contents and

verification of the child elements has been carried out through the use of Schema_ocr.xsd The final output received from the OCR system is encoded using XML tags. Finally after the completion of all the modules of Oriya OCR we are able to convert the OCR output text into HTML format.

ACKNOWLEDGMENTS

We acknowledge the financial and technical support from D.I.T, M.C.I.T, Government of India for the work.

REFERENCES

- 1 World Wide Web Consortium, Namespaces in XML, January 14, 1999
- 2 Jessica L. Glace and Mark R. Crawford "Recommended XML Namespace for Government Organizations". GS301L1/AUGUST 2003.
- 3 James Clark. "XML Namespaces", Mechanism specified by the W3C XML Namespaces Recommendation. February 02, 1999.
- 4 Tim Bray "XML Namespaces by Example", January 19, 1999.
- 5 Houlding, S.W.: Xml – an opportunity for meaningful data standards in the geosciences. Computers & Geosciences 27(7) (2001) 839 – 849.
- 6 C. V. Jawahar and Anand Kumar: Content Level Annotation of Large Collection of Printed Document Images. In: Proc. of International Conference on Document Analysis and Recognition (ICDAR). (2007) 799–803.
- 7 C. V. Jawahar, Anand Kumar, A. Phaneendra, K. J. Jinesh: Building Data Sets for Indian Language OCR Research. Springer Series in Advances in Pattern Recognition (2009).
- 8 W3C Web Ontology Working Group: Web Ontological Language (OWL). <http://www.w3.org/TR/owl-guide/> (2004)
- 9 Gaurav Harit, K. J. Jinesh, Ritu Garg C.V Jawahar, and Santanu Chaudhury Managing multilingual OCR project using XML. Source, ACM International Conference Proceeding Series
- 10 Federal CIO Council XML Working Group, Draft Federal XML Developer's Guide, April 2002.
- 11 T. Berners-Lee, R. Fielding, L. Masinter; Internet Engineering Task Force (IETF) RFC 2396, Uniform Resource Identifiers (URI): Generic Syntax; Internet Society; August 1998. archive. Proceedings of the International Workshop on Multilingual OCR, Barcelona, Spain, 2009

Upcoming Standards Related to Graphics, Data Storage and Script Processing in Future Web Applications

Shwetank Dixit

Opera Software ASA, Waldemar Thranes Gate 98, Oslo, Norway
shwetankd@opera.com

Abstract— HTML5 is one the verge of making web applications more powerful and easier to code. However, there are other W3C standards which are also, in their own way, trying to make web applications more powerful, aesthetically better and easier to code for. Hence it makes sense to take a look at how all these standards could be used by developers, and what it could mean for the future of the web. This paper will cover other upcoming standards such as CSS3, W3C Web Storage, Web SQL DB, Web Workers API, Canvas 2D API and others and see how these could work with HTML5 and its features in order to make the next generation of web applications.

Keywords— CSS3, HTML5, Web Storage, Offline Applications, Web Storage, Canvas

I. INTRODUCTION

HTML5[1] is one the verge of making web applications more powerful and easier to code. However, there are other W3C standards which are also, in their own way, trying to make web applications more powerful, aesthetically better and easier to code for. We will look into some of the W3C Specifications and see what they are and how they can effect the future of web applications.

II. ADVANCEMENTS IN STYLING OF WEB PAGES USING CSS3

CSS3[2] is the further advancement of the Cascading Style Sheet specification. The work on CSS3 is modularised into different parts. These parts will be separate recommendations in themselves.

A. CSS3 Web Fonts

CSS3 Web Fonts[3] provide a way to use custom fonts on web pages. Till now, user agents only supported six font types for web pages across platforms. CSS3 Web Fonts provides a way to display text in a web page using any font in .ttf or .otf format. The advantages of this approach include better aesthetic matching of the text to the general design of pages, and also better readability in certain scenarios, especially in the case of non-latin fonts.

CSS3 Web Fonts are introduced using an '@font-face' property. An example of it would be

```
@font-face {src: url('sampleFont.ttf');font-family: 'myFont';}
```

And then using it in the same way as a normal font, such as

```
#samplediv{font: 2.5em 'myFont';}
```

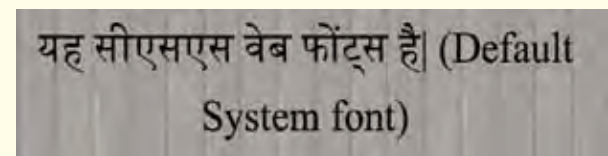


Fig. 1. Example of Hindi text in web page using the default system font available on the Mac OSX platform

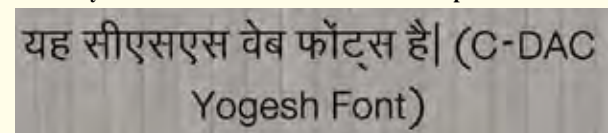


Fig. 2. Example of Hindi text in a web page using C-DAC Yogesh Font, using CSS3 Web Fonts.

As mentioned earlier as well, this can be especially useful in presenting non-latin based languages in a more readable format. We shall take the example of Hindi.

Figure 1 shows hindi text in a web page using the default system font available. Figure 2 shows the hindi text using CDAC Yogesh font (not available on the system by default) and embedded in the page using the @font-face property of CSS3 Web Fonts.

B. Rounded Corners with CSS3 'border-radius' Property

'border-radius'[4] is part of the CSS Backgrounds and Borders Module Level 3 Candidate Recommendation[5] by the W3C. A part of a page, usually an <div> element, having rounded corners, is a very common effect used by developers. However, so far, it has been done usually by using images and using multiple nested <div> tags. Not only does it unnecessarily increase the complexity of the code, but also consumes greater bandwidth because of the images being used.

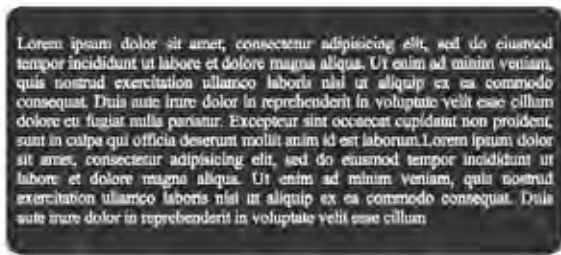


Fig. 3. Example of div element in a web page using rounded corners generated through CSS3 border-radius property.

It works as:

border-radius: p, q, r, s;

Where p , q , r and s are the top-right, bottom-right, bottom-left and top-left border radii respectively. Many browser vendors currently support this property through their own specific vendor prefixes (-o, -moz and -webkit prefixes), however, the eventual aim is to move away from vendor prefixes and have developers use one standard 'border-radius' property.

C. Opacity and Precision Colour Control

CSS Color Module Level 3[6] provides the ability to control the opacity[7] of elements as well as have precise control over their colour.

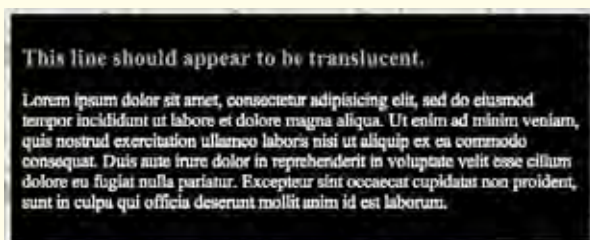


Fig. 4. Example of an <H2> header element in a web page appear translucent using the 'Opacity' property.

It works as:

opacity: x

Where x is a numerical value between 0.0 and 1.0.

Colour control is improved by providing two mechanism, namely, RGBA[8] and HSLa[9]. RGB corresponds to the Red-Blue-Green colour model and HSL corresponds to the Hue-Saturation-Lightness colour model. the 'a' in both corresponds to the alpha channel which can give added opacity control to the colours.

In both RGBA and HSLa, values can be defined either as integers, or as percentages. However, both cannot be used in combination to define a value.

This box should appear to be translucent.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Fig. 5 Example of div element in a web page appear translucent and consequently a partial view of the background image being present.

An example of RGBA and HSLa would be:

color: rgba(100%, 50%, 20%, 0.9);

color: hsla(80%, 70%, 60%, 0.5);

The ability to add such colour controls makes it possible for designers to have a precise control over their designs.

III.WEB STORAGE AND OFFLINE APPLICATIONS

A. W3C Web Storage API

The W3C Web Storage API[10] is designed to be a better way to store data on the client side. It makes it possible to store key-value pairs on the client side. It has two parts -Local Storage and Session Storage.

1) **Session Storage:** Session: Storage[11] is storage meant only for that particular session. As soon as the browser tab or window is closed, all data stored in Session Storage is supposed to be lost. This is useful for applications such as shopping sites etc where highly sensitive transactions take place, which should not be reproduced by opening the tab again

2) **Local Storage:** Local Storage[12] is meant to be a persistent form of storage. It is meant to store value persistently, which means that if the browser tab or window is closed, but re-opened (or if the user goes to that same page again), the information should still be available.

Key-Value pairs are stored in Session and Local Storage as: `sessionStorage.setItem(yourkey, yourvalue);` and `localStorage.setItem(yourkey, yourvalue);`

Data is retrieved for session storage, by: `var data = sessionStorage.getItems(yourkey);` and for local storage by: `var data = localStorage.getItem(yourkey);`

The specification also mentions a storage event to be fired whenever there is change to the storage present. With it, it is possible to know the nature of the storage, the previous and new value of the storage, the URL of the page whose key has changed, as well as the key itself.

It is also interesting to note that there is another specification, called the W3C Web SQL Database Specification[13], which specifies how to store, retrieve and manipulate data on the client side using SQL like queries. However, as of now, that specification is in a state of impasse, as there has not been a clear consensus on how to support the SQL dialect, which has been left as a reference to that of SQLite's, so far.

B. HTML5 Offline Applications

The HTML5 specification also deals with how applications could work offline. The specification outlines something known as 'application cache'[14]. Under this, the files needed for the application to run offline are referenced in a file called a 'manifest file'. This manifest file can tell the browser which files to cache for offline use. These files can be anything which is required to run it, i.e, images, pages, JavaScript files, videos, CSS files, etc.

Much of the application cache API is non-normative, however, it has laid out certain constants to describe the current state of the application cache ('idle', 'checking', 'downloading', etc), which are returned when invoking 'cache.status', which tells the status of the application cache.

Application Cache (especially, if combined with a persistent client side data storage mechanism like Web Storage or Web SQL DB) can be very useful in providing the web application offline, when the user does not have connection to the internet. In particular, mobile devices can benefit from it, as in many cases, signal might be low or nil, but the user might want to still access the web application.

IV. OTHER STANDARDS AND PROPOSALS

A. Web Workers

Web Workers is[15] a draft by the W3C which outlines how JavaScript, usually relatively expensive JavaScript in terms of computation and/or time, could be done on the background, in different 'workers', in a thread-like way. In other words, it provides a mechanism for performing background scripts without interrupting other scripts such as those for clicks and

other user interaction elements.

It is possible to create a worker to run from a script, which could be an external file. Also, it is possible to create a shared worker, so that requests from multiple files could be handled.

A new worker can be created as follows:
new Worker("worker.js")

Where 'worker.js' contains the script needed to be executed as a worker. However, stipulations include the fact that workers do not have access to the DOM (Document Object Model) of the page, and neither do they have direct access to the parent page[16]. The worker is supposed to be strictly for computation, and not designed for user interactive behaviour. Communication between the worker and the page is done using the postMessage API[17] which is part of HTML5 Web Messaging[18]. The ability given by Web Workers to process code in the background could be very useful in complicated web application which require heavy JavaScript computation.

B. Canvas 2D API

The <canvas> tag[19] is part of the HTML5 specification. However, the API required to make graphics with it have been moved to a separate specification, called the Canvas 2D API [20]. The <canvas> tag is basically similar to an empty element, except in this, you can program the pixels using scripting. The API to do this, is specified in this API. This makes it possible to draw programmable 2D graphics, by making use of this API and JavaScript.

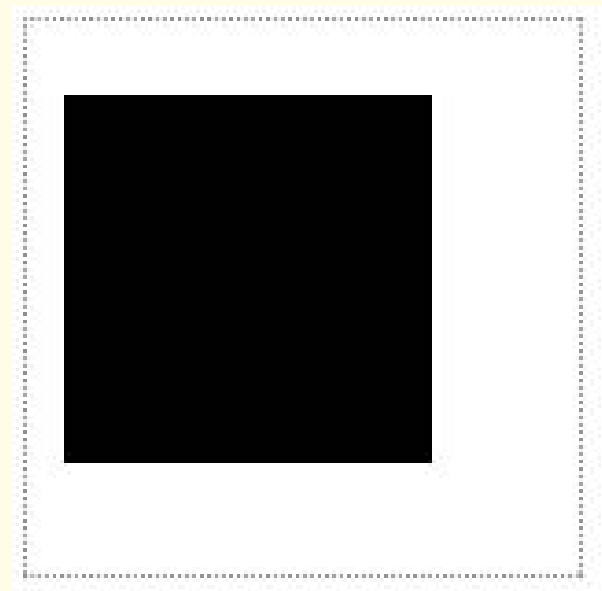


Fig. 6. Example of gradient drawn with the canvas element and API.

To draw using canvas, the first thing to do is to get the context using the `getContext` method

For example:

```
context = canvas . getContext(contextId)
```

Basic shapes like ellipses and rectangles (which also means, in effect, circles and squares) can be made, as well as arcs, lines and curves. The API provides ability to draw quadratic curves as well as bezier curves.

For example, a simple rectangle can be coded by writing:

```
context.fillRect(0,0,50,100)
```

Besides drawing of shapes, various effects such as transforms, rotating, scaling and translating are also provided in the API.



Fig. 6 Example of gradient drawn with the canvas element and API.

Specifying colours, alpha channels, gradients and patterns are also available. 2D Graphics made with the help of this API can be made interactive and action driven, by combining them to work with user driven events, such as on 'onclick' event handler.

V. CONCLUSIONS

Future web application would in all likelihood be using HTML5 in combination with other upcoming standards and technologies. These standards have their own use specific and unique uses, providing the ability for future web application to have graphics in a better and easier fashion, better readability for many languages on the web, for storage of data on the user's side and for offline web application, as well as for processing of intensive JavaScript methods concurrently in a thread-like manner.

REFERENCES

- [1] *HTML5 Working Editor's Draft*, W3C, 2010.
- [2] *Introduction to CSS3: Working Draft*, W3C, 2001.
- [3] CSS Fonts Module Level 3 Available: <http://www.w3.org/TR/css3fonts/>
- [4] The 'border-radius' Property. Available: <http://www.w3.org/TR/css3background/#the-border-radius>
- [5] *CSS Backgrounds and Borders Module Level 3: W3C Candidate Recommendation*, W3C, 2009
- [6] *CSS Color Module Level 3: W3C Working Draft*, W3C, 2008
- [7] Transparency: the 'opacity' property. Available: <http://www.w3.org/TR/css3-color/#transparency>
- [8] RGBA Color Values. Available: <http://www.w3.org/TR/css3-color/#rgba-color>
- [9] HSLA Color Values. Available: <http://www.w3.org/TR/css3-color/#hsla-color>
- [10] *Web Storage: W3C Editor's Draft*, W3C, 2010
- [11] The Session Storage Attribute. Available: <http://dev.w3.org/html5/webstorage/#the-sessionstorage-attribute>
- [12] The Local Storage Attribute. Available: <http://dev.w3.org/html5/webstorage/#the-localstorage-attribute>
- [13] *Web SQL Database: W3C Editor's Draft*. W3C, 2010
- [14] Application Caches. Available: <http://dev.w3.org/html5/spec/Overview.html#appcache>
- [15] *Web Workers: W3C Editor's Draft*. W3C, 2010
- [16] Computing with JavaScript Web Workers. Available: [Computing with JavaScript Web Workers](http://dev.w3.org/html5/spec/Overview.html#appcache)
- [17] The `postMessage` function. Available: <http://dev.w3.org/html5/postmsg/#dom-window-postmessage>
- [18] *HTML5 Web Messaging*, W3C, 2010
- [19] The Canvas Element. Available: <http://dev.w3.org/html5/spec/Overview.html#the-canvas-element>
- [20] *Canvas 2D API Specification 1.0*, W3C, 2009

Smarter Browsers for Smarter Web

Tathagata Bhattacharjee¹, Nidhi Patharia²

¹Department of Advanced Software and Computing Technology,
International Institute of Information Technology, P-14 Rajiv Gandhi Infotech Park,
Hinjewadi, Phase – I, Pune – 411057

²Department of Advanced Software and Computing Technology,
International Institute of Information Technology, P-14 Rajiv Gandhi Infotech Park, Hinjewadi,
Phase – I, Pune – 411057

¹tathagatab@isquareit.ac.in ²nidhip@isquareit.ac.in

Abstract— The web is a platform on which people can share data, knowledge or information easily and fast. But this sharing is always confined to certain boundaries of the application span. Like, if we want to compare cost of a product in the web, a site gives us the pricing snapshot of that time, or we need to fetch multiple snapshots over a time and compare them to derive cost change patterns. i.e., a human interaction in this process of comparison is required.

Semantic web is a web with meaning. So for any comparison we should naturally be able to derive and fetch meaningful data from various snapshots at different timeframes.

Smarter browsers would derive meaning for our searches by knowing our past trends. The browser should be able to link our existing comparisons, to the comparisons we have done in past and present the comparative information accordingly. This would help us to actually compare longitudinal data over a timeframe to our context.

Our case study is based on the Health and Demographic Surveillance System data that are collected at different geographic locations at regular intervals and hence longitudinal in nature and sharing with comparisons at different timeframes is a necessity

Keywords— Semantic Web, Browser, URI, RDF, OWL, Longitudinal Data

I. INTRODUCTION

The word semantic means study of meaning in general. But in the context of web the word “Semantic” has a greater significance. The aim of semantic web is to add intelligence to the ubiquitous web. The Semantic web is the web of data with meaning in the sense that a computer program can learn enough about what the data means to process it, i.e. a web where machines can read sites as easily as humans read them. That means, presentation layer is separated from the data layer so that the information hidden in different websites can be presented to the user in a way he desires and hence

better inferences can be made.

The World Wide Web has revolutionized the way distant computers interact with each other so that individuals can collaborate and communicate without regard to their geographical locations. Like most of the social networking sites that connect people to others they know or share common interests.

Web 1.0 the first generation of the web was primarily used, to give information. Web 1.0 began with the release of the WWW to the public in 1991, and is the general term that has been created to describe the Web before the world saw web 2.0 in 2001, which has changed the definition of web altogether. The only contributors to web 1.0 were the creators of the web page. Web 1.0 was about reading whereas web 2.0 is more about writing. Web 1.0 was about information whereas web 2.0 is more about opinion. It is the user of the site who contributes more to the dynamism of the site.

Semantic web is not something out of the box. It is more of an extension to the already existing web, in which just the information would not be enough. The web must be able to define the meaning of the information and services so that not only humans but computers can actually draw inferences from the web content. The Semantic Web envisages itself as “web of data”; hence it not only puts together the large volume of related data on the web, but also connects that information with data in relational databases and other non-interoperable repositories.

So what is this hype about? Data that is generally hidden away in HTML files is often useful in some contexts, but not in others. The problem with the majority of data on the Web which is in this form at the moment, is that, it is difficult to use on a large scale, because there is no global system for publishing data in such a way as it can be easily processed by anyone. For example, just think of information about historical events, sports, weather information, online dictionaries, etc, all of this information is presented by numerous sites, but all in HTML. The problem with such data presentations is that it is difficult to use this data in the way one might want to do so.

With Semantic web it would become easier to publish data in such a way that can be used for a variety of different tasks. The Semantic Web is generally built on syntaxes which use URIs to represent data, usually in triples based structures: i.e. many triples of URI data that can be held in databases, or interchanged on the World Wide Web using a set of particular syntaxes developed especially for the task. Thus to put it formally, the element of Semantic Web include Resource Description Framework (RDF) and a variety of data interchange formats (e.g. RDF/XML, N3, Turtle, N-Triples), and notations such as RDF Schema (RDFS) and the Web Ontology Language (OWL). All these provide a formal description of concepts, terms, and relationships within a given knowledge domain.

The RDF language is a part of the W3C's Semantic Web Activity. RDF is a framework for describing resources on the web which can be read and understood by computers and is written in XML. The people seeing these resources do not see RDF.

OWL (Web Ontology Language) is a language for processing web information. Ontology is about describing the things and their relationships. OWL is built on top of RDF to process the information on the web.

The web has evolved from providing information to becoming intelligent enough to understand the user's requirement. But one thing that remains common in the evolution of the World Wide Web is single point of access that retrieves, presents, and traverses information resources on the World Wide Web. – The Web Browser.

II. APPROACH

For the smarter web (semantic web), where data is in presentation free format, the browser should also be smarter enough to read it, and interpret according to the application requirements. In our approach we concentrate on timelines. This approach helps the users to actually compare the information of use to them over a period of time to make better comparative study.

The browser reads the RDF (Resource Description Framework) tags and the OWL tags and presents the useful information to the user from multiple web resources based on the relations between them. E.g. if a user searches for flights to New Delhi, the web should be able to understand that he wishes to visit New Delhi, so tries to find out other things that he would like to know before he reaches there, like the weather forecast, the tourist places, political stability, directions to important landmarks etc. The information

may be available on the internet but not necessarily at one place. So the web picks up all this information which is disseminated on different sources and presents it to the user.

But what if the user needs a comparison over a period of time? Here the browser must act smart and provide a comparative vision of the information previously saved on his machine. The browser must be able to identify the relationships between the previously saved information and the new information the user views. Browser must also understand the presentation preferences of the user and arrange the snapshots of the similar information over a period of time according to his choices and allow him to share it for peer group access. If the information he has saved becomes irrelevant after a period of time the browser must allow him to delete it from his repository.

III. ARCHITECTURE

Fig 1 shows the proposed application architecture where the web browser communicates with the semantic web to get cumulative information from disseminated sources. The browser communicates with a local application that differentiates the presentation layer from the data layer. The presentation layer understands the users display preferences and the data layer stores the RDF tags and the date & time when the information was bookmarked. This application identifies similar tags and decides the action. This application makes the browser smart enough to understand the timelines and find out the relationships that exist between the resources over a period of time.

The user may choose to publish this collective information to a publishing repository which provides other users the access to this comparative study who may in turn wish to add/update/delete certain

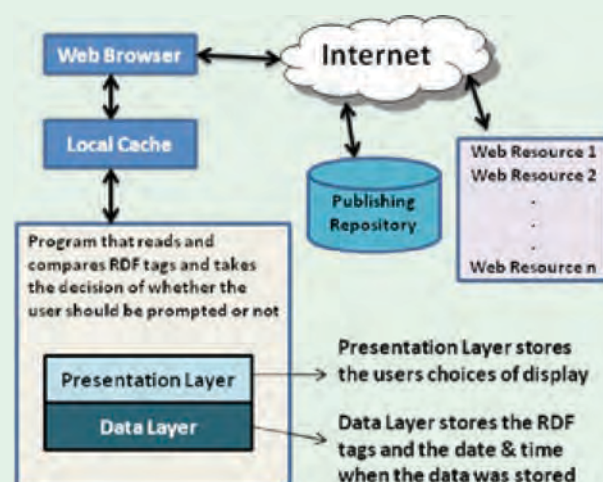


Fig. 1. Architecture

information according to their needs and publish it again if they want to, to the same repository.

IV. PROCESS FLOW

The process flow is represented in fig. 2. The user may visit a web resource directly or search for it. He chooses to bookmark the page displayed to him. The browser saves a snapshot of the information & the time. The user can publish this to the publishing repository. If the user views the similar information over a period of time, the application compares the matching RDF tags and prompts the user that he had already viewed the similar information on a certain date and the presentation layer arranges it in form of different window snapshots on the same page. Now when the user sees a comparison between the similar information from different web resources over a period of time, he may choose to save/ discard the unsaved information. If he chooses to save it, the tags and their relationships are added to the application's repository so that the user sees this information the next time he looks for similar information.

The user also has the facility to publish the comparisons on a web repository and share it among peer groups. Peers who view this published result, may also add/delete/modify the information according to his requirements or likings, and re-publish differently and share it with peer groups.

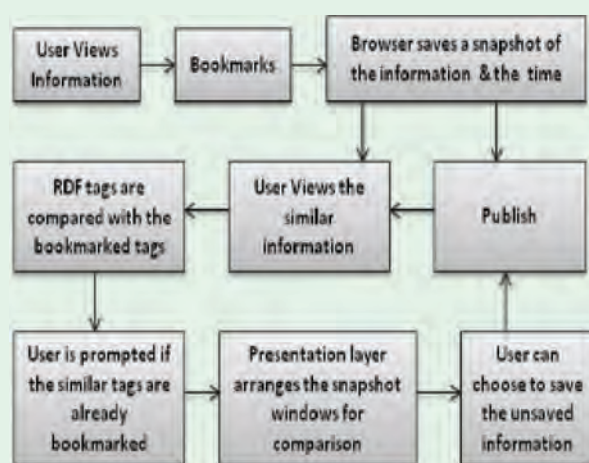


Fig. 2. Process Flow

V. A CASE STUDY

We explain our approach on a real life case study, on longitudinal data of Health and Demographic Surveillance Systems (HDSS).

Health and Demographic Surveillance Systems collect longitudinal data within designated population

area over a period of time. This data after processing shows characteristics of human population within the area. A comparative study amongst many such areas over different timelines provides a vivid population dynamics and patterns globally.

The application helps the users to not only share and compare data patterns from within a specific population area, but globally from any web resource offering such information. Once a user looks for a specific type of information like population pyramid, crude death / birth rate, in / out migration, the semantic web would gather information from all similar sites and provide a comparison option.

E.g. a scientist who doesn't know much about semantic web visits one of the resources that provides Demographic data of 2009 to see the population distribution and the deaths due to heart attacks in a particular region. He bookmarks this information in his local machine. After some time he sees the similar information for the year 2010. The application would compare the matching tags and the relationships between them and prompt the user that he saved the same information sometime ago and the trends may have changed. The user looks at the pages he bookmarked as a summary along with the date & time.

This type of information when provided altogether at one place that highlights the differences between the population trends now and then helps the user make better inferences without actually bothering about the technology behind the scenes. Hence the user may concentrate more on his research than looking for pieces of information distributed in different sources.

Fig 3 shows the snapshot of the browser window the user would see when he looks at similar information over a period of time. He sees the snapshots of the resources he saw on certain dates altogether on a single window. The user may now choose to compare, publish or delete all/some part of this. If he chooses to publish the selected snapshots along with their details are published to the repository where the peers can access it. But if the user wishes to compare the details he sees something like in Fig 4. This type of representation helps the user to actually see the summary of the information which may help him to look at all the perspectives of it over a period of time. He can easily identify the changing trends and make better inferences. If he thinks that certain part may not be needed anymore he may select it and delete it.

In the fig 4 the snapshots with green title bars indicate the saved information and their titles indicate the date when they were saved. The ones with red title

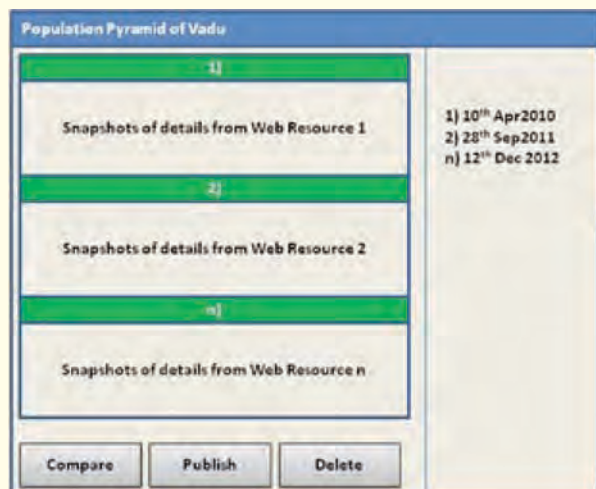


Fig. 3. Browser Snapshot

bars indicate the unsaved information user is seeing at present. The user may choose to save it or discard it according to his needs.



VI. CONCLUSION

To conclude, we would like to infer that the web should provide meaningful data to the user by understanding user's dynamic requirements. At present, most sites present data to the user, which is the easier task, and the user interprets and creates the relationship to extract meaning, which is more difficult. Semantic web can reverse this, and can make the machine do the interpretation meaningfully, and present information to the user intelligently as he wants.

Along with the web the browser must become smart enough to read and present this information intelligently so that the user gets the maximum benefit out of it.

ACKNOWLEDGMENT

We acknowledge the support extended to us for this initiative by the Department of Advanced Software and Computing Technologies and management of the International Institute of Information Technology, Pune.

REFERENCES

- [1] Dennis Quan and David R. Karger, *How to Make a Semantic Web Browser*, p.413-422, November 02-06, 1999, Kansas City, Missouri, United States.
- [2] Phillip Green., *Web 3.0: Rise of the Intelligent Machines*, Information Management Special Reports, September 1, 2009
- [3] What "Web 3.0" Will Mean to You, A Cision Executive White Paper
- [4] Tim Berners-Lee, "Semantic Web Roadmap," w3.org., September '98 [Online]. Available : <http://www.w3.org/DesignIssues/Semantic.html>
- [5] Naveen Balani, "The future of the Web is Semantic", ibm.com, October 18, 2005 [Online] Available: <http://www.ibm.com/developerworks/web/library/wa-semweb/>
- [6] David Huynh, Stefano Mazzocchi, David Karger, *Piggy Bank: Experience the Semantic Web Inside Your Web Browser*, Lecture Notes in Computer Science, Volume 3729, Oct 2005, Pages 413 – 430
- [7] G Cormode, B Krishnamurthy, "Key Differences between Web1.0 and Web2.0", January 2008 [Online]. Available: <http://www.mendeley.com/research/key-differences-between-web10-and-web20/>
- [8] Eric Miller, "An Introduction to the Resource Description Framework", May 1998, [Online] Available: <http://www.dlib.org/dlib/may98/miller/05miller.html>