Context-Aware Interaction Approach to Handle Users Local Contexts in Web 2.0

Mohanad Al-Jabari[1,⋆], Michael Mrissa[2], and Philippe Thiran[1]

[1] PReCISE Research Center, University of Namur, Belgium
[2] Université de Lyon, CNRS
Université Lyon 1, LIRIS, UMR5205, F-69622, France

Abstract. Users sharing and authoring of Web contents via different Web sites is the main idea of the Web 2.0. However, Web users belong to different communities and follow their own semantics (referred to as local contexts) to represent and interpret Web contents. Therefore, they encounter discrepancies when they have to interpret Web contents authored by different persons. This paper proposes a context-aware interaction approach that helps Web authors annotate Web contents with their local context information, so that it becomes possible for Web browsers to personalize these contents according to different users' local contexts.

1 Introduction

Web users usually belong to different communities and follow their local contexts for representing and interpreting Web contents. A *local context* (or context, for short) refers to the shared knowledge of a community such as a common language and common local notations and conventions (keyboard configurations, character sets, and notational standards for measure units, time, dates, durations, physical quantities, prices [3,4,14]. As a consequence, the same real world concepts might be represented and interpreted in different ways by different *Web authors and readers*. Such concepts are referred to in the following as Context-Sensitive Contents, or *CSCs*). For example, the concept of *price* could be represented using different currencies (e.g., Euro, US Dollar) and according to different price formats. Also, *date and time* concepts could be represented using different time zones and according to different formats. This situation leads to several *discrepancies* Web readers encounter on the Web, as they (need to) follow their contexts to interpret these *CSCs*.

Recently, the Web 2.0 has revolutionized the way information is designed and accessed over the internet. In our previous work [2], we advocated that the heart idea of the Web 2.0 lies into sharing and authoring of Web contents via different Web users and sites. Also, we illustrated a set of use cases that Web 2.0 sites/services provide. Indeed, Web 2.0 sites enable users not only browsing the Web but also creating and updating Web contents (i.e., they can act

⋆ Supported in part by the Programme for Palestinian European Academic Cooperation in Education (PEACE).

B. Benatallah et al. (Eds.): ICWE 2010, LNCS 6189, pp. 248–262, 2010.

as *Web authors*). Several authors can author and update the same Web contents (e.g., *wiki*). In addition, Web 2.0 sites/services can aggregate Web contents from several sites and display them as a single Web page. Contents aggregation may happen both on client-side (e.g., using RSS) or on server-side (e.g., blogs' aggregations on sites like *Technorati*). Therefore, we concluded that Web contents in a *single Web page* are represented according to *several* authors' contexts, and the discrepancies Web readers encounter increase accordingly.

One possible solution is to annotate *CSCs* with semantic metadata (i.e., authors' contexts), so that it becomes feasible for Web browsers to adapt the former to different users' contexts [6,10]. This paper proposes a context-aware approach to resolve the discrepancies Web users encounter when they interact with Web 2.0 sites. This approach is an extension to our previous work [2,3] and consists of: (1) an evaluation of several design alternatives to adapt *CSCs* to different users' contexts; (2) a model that describes how *CSCs* are annotated with context information; (3) a context-aware architecture that shows how our approach works seamlessly with Web technologies; and (4) an annotation process that details how Web authors are assisted to specify their contexts and to annotate *CSCs* with a suitable context information.

The rest of the paper is structured as follows. Section 2 introduces a motivating example and evaluates the design alternatives. Section 3 summarizes a semantic model proposed in [3] to represent *CSCs* together with context information. Section 4 illustrates the annotation of *CSCs* based on the semantic model. Section 5 presents our proposed architecture and Section 6 details the annotation process. Section 7 introduces a prototype as a proof-of-concept. Section 8 discusses related work, and Section 9 concludes the paper.

2 Motivation and Design Alternatives

2.1 Motivating Example

This section presents an example to illustrate the discrepancies that Web users could encounter when they interact with Web 2.0 sites, as shown in Figure 1. This example considers several authors and readers from different communities. Also, it considers several tasks (i.e., T1-T7) performed on different Web 2.0 pages in sequential manner as follows:

- A British author creates and publishes a length and a date contents on page A (T1). After that, an American author browses the contents of page A (T2), and then updates the date content created by the British author to *07/09/2009* and publishes it again (T3).
- A Canadian author (from French speaking community) browses the contents of page B and deletes the date content *2009-09-11* (T4). We consider the page B's contents were created by this author. Next, the length and date contents from pages A and B are aggregated to page C (T5).
- An Italian reader browses the date contents that are automatically aggregated, via RSS engine, from pages A and B (T6). Finally, a French reader browses the date and length contents that are aggregated to the page C (T7).

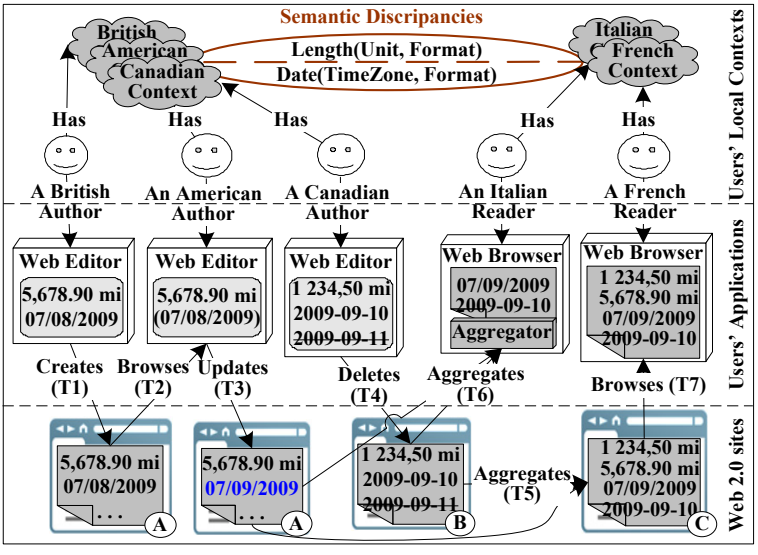

Fig. 1. Web 2.0 contents' sharing and the implicit use of users' local contexts

It is obvious that the date and length contents are represented in different ways by different authors. For example, the British author implicitly uses the British context[1] in T1. In contrast, Web readers usually (need to) interpret these contents according to their contexts. For example, as the French reader uses the Meter unit and the French length format (e.g. 1 234,50), he is responsible to adapt the length from Mile to Meter and to French length format. The problem is similar with respect to the date content 07/09/2009 which is updated at task T3. It is not obvious whether the American author updates the date content according to his context or according to the British context of the original author. Even if the ambiguity is resolved (i.e., he uses his context), the French reader might misinterpret this date as the 7^{th} of September (following the French format) instead of the 9^{th} of July (following the American format). Finally, several time zones are implicitly used by different users for representing and interpreting the date contents.

To conclude, the local context is clearly part of the $CSCs'$ semantics. Also, the discrepancies that rise do not relate to the $CSCs$ themselves, but rather to the contexts of Web users who represent and interpret them.

2.2 Design Alternatives

To resolve the aforementioned discrepancies, there is a need to adapt $CSCs$ from their multiple authors' contexts to their readers' contexts. To do so, there are several design alternatives, each of which has its own strengths and limitations. In the following, we evaluate three alternatives with respect to Web 2.0 use cases.

[1] Mile unit, British notation (e.g. 1,234.50) and date format (dd/mm/yyyy).

Adaptation to a Standard Local Context

The first alternative imposes a standard, unified context for all Web sites. Then, each *CSC* needs to be annotated with a standardized machine interpretable version (*MV*). The latter is generated by adapting the value of *CSC* from the author's local context to the standard context at creation and update time. Additionally, there is a need to adapt the *MV* into different human-readable versions according to different readers' contexts. In practice, we can use Microformat specifications and Microformat's *abbr design pattern*[2] for generating *MV* and annotating *CSC*. For instance, the date content above can be annotated with an *MV* date based on the ISO 8601 date/time standard.

This alternative allows *CSCs* from several sites to be aggregated seamlessly as they are annotated with unified *MV*s. However, it violates the "do not repeat yourself" (DRY) design principle [1]. Indeed, each *CSC* needs to be represented twice (in the text and in the *MV*), and therefore needs to be maintained twice. For instance, when an author updates an annotated *CSC*, then both versions need to be updated. In addition, it lacks flexibility and may not satisfy the requirements of all Web sites. For example, most Europe countries use a tax system called VAT, while different states in the USA use different tax systems. Finally, *CSCs* need to be adapted twice: one from the author's version to *MV* version and from the latter to the reader's version.

Adaptation to a Single Page Local Context

The second alternative is to identify a local context for each page. In this setting, each *CSC* is adapted from an author's context to a page's context at creation and update time, and adapted to different readers' contexts at browsing time.

This alternative does not violate the DRY principle and does not impose a standard context. Moreover, Web contents in a single Web page are homogeneously represented. However, *CSCs* need to be adapted many times. These adaptations are often not necessary. For instance, assume the British author above needs to update the date content he created before. To this end, this date needs to be adapted to the author's context, since it was adapted from the author's to the page's A contexts at creation time. Also, it needs to be adapted from the author's to the page's A contexts after update. Furthermore, when the date and length contents are aggregated from pages A and B to the page C, then other unnecessary adaptations are needed to adapt the aggregated contents according to the context of page C.

Annotation of a *CSC* with Author's Local Context

The final alternative is to annotate *CSCs* with authors' contexts at creation and update time and to adapt the annotated *CSCs* to the readers' contexts at browsing time. For instance, annotating the above date and length contents with several authors' contexts, and adapting them to the French contexts at task T7.

[2] See `http://microformats.org/wiki/`

This alternative does not violate the DRY principle and does not impose a standard context. Furthermore, it preserves the initial Web contents as they were submitted to the Web page, which may be useful for their in-depth understanding or analysis. Also, it optimizes the number of required adaptations of *CSCs*. Finally, context information like *date format* can be made visible to readers in case *CSCs* cannot be adapted to readers' contexts. However, when an author needs to update a *CSC* created by another author, the Web editor must take care to update the annotation too (hidden from the user).

2.3 Discussion

Our proposal is to adopt the third design alternative, since it is the best tradeoff with respect to the context representation flexibility, the DRY principle, and the number of *CSCs* adaptations, as summarized in Table 1 below:

Table 1. Evaluation summary of the design alternatives

Design Alternatives	**Standard Context**	**Single Page's Context**	**Multiple authors' Contexts**
Context Rep. Flexibility	No	Yes	Yes
DRY Principle Compliance	No	Yes	Yes
Number of Adaptations	Twice	Many	Once

However, there are several techniques to annotate *CSCs*, and annotation of *CSCs* is still a complex process. Indeed, we should consider that authors often do not know the relations between *CSCs* and local context information. They also do not have theoretical and technical knowledge about the annotation process. As a consequence, we identify several objectives to be addressed in the rest of the paper as follows:

1. Identify the relations between *CSCs* (e.g., date) and context information (e.g., date format and time zone) at the conceptual (meta) level.
2. Evaluate and optimize different annotation techniques, and illustrate how to annotate *CSCs* with context information.
3. Assist authors (e.g., British author) to specify their contexts and annotating each type of *CSCs* (e.g., date) with suitable context information (e.g., date format ="dd/mm/yyyy", date style ="short", and time zone ="UTC").

3 Semantic Model of Web Contents

To address the first objective presented in Section 2.3, we proposed a semantic representation model in our previous work [3]. This model builds on a local context ontology and uses the notion of semantic object to represent each type of *CSC* together with suitable context information, as summarized in the following.

3.1 Local Context Ontology

As already mentioned, *CSCs* such as date, time, price, physical quantity, phone number, and address are represented and interpreted in different ways by different users. To annotate *CSCs* with authors' context, we specify a set of local context attributes in an ontology called *local context ontology*. These attributes are mainly related to *country and community*. Indeed, each country has a set of local conventions such as currency, tax, measure system, etc. Also, each country has many cities, sometimes located in different time zones. In addition, one country may have one or more communities (e.g., French and Dutch speaking communities in Belgium). Each community usually relies on a common natural language and a set of common conventions such as writing formats of the above *CSCs*. More details about local context ontology and *CSCs* are given in [3].

3.2 Semantic Object

A *semantic object* provides a way to represent each *CSC* together with one or more context attributes. Basically, a semantic object *SemObj* is a triple $\langle S, V, C \rangle$. S represents the real world concept that the *CSC* adheres to, V is the physical representation (the value) of the *CSC*. C specifies the local context of *SemObj*. This context is represented as a finite set of context attributes. Also, context attributes themselves are represented as semantic objects, which may also have context attributes. This provides a recursive means for context description. Also, we categorized context attributes into two subsets: static and dynamic. Static attributes are the minimum context attributes that are used to describe the context of a semantic object and hence, their values must be specified explicitly. Dynamic attributes can be inferred from other context attributes that belong to that semantic object (See Table 2 below). However, Web authors can explicitly specify the values of one or more dynamic attributes if required, thus overriding the inference results. The motivation behind dynamic attributes is twofold. First, some dynamic attributes such as currency exchange rate could have dynamic value, and therefore cannot be statically stored. Second, it simplifies the specification of context information. For example, it is easier for users to specify the city instead of the time zone of this city.

To illustrate the notion of semantic object, Figure 2 represents the date from our scenario updated by the American author during Task T3 as a semantic object (See Figure 1). *Date* refers to the *date* concept S. *'07/09/2009'* represents the value V. Finally, *Context* represents the set of context attributes C. Here, the date *SemObj* has *DateStyle* as *static* attribute, and the *DateFormat* and *TimeZone* as *dynamic* attributes. The other context parts further describe the context of other semantic objects (i.e., DateFormat and TimeZone). The *DateFormat* value is inferred from the *country*, *language*, and *dateStyle* and the *TimeZone* value is inferred from the *country* and *city* static attributes.

Table 2 summarizes the relations between *CSCs* and static/dynamic context attributes. These relations are mainly derived from the *W3C Internationalization* initiatives that helped us to build *CSCs* [9]. As we will see in Section 6, these relations are utilized to extract the context attributes of semantic objects.

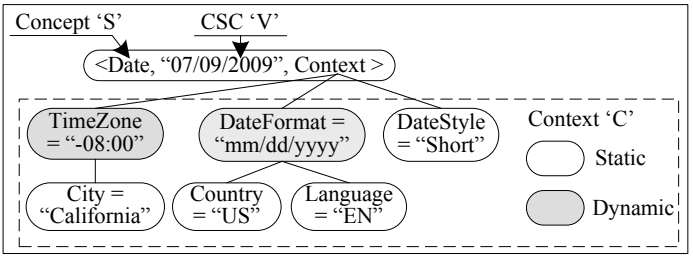

Fig. 2. Sample of date semantic object

Table 2. Relations between *CSCs* and context attributes (the value of a dynamic attribute can be inferred if the value(s) of attribute(s) between brackets are known)

Context-Sensitive Contents (*CSCs*)	**Context Attributes**	
	Static	**Dynamic**
Date/Time	Date style	Date format(Country, Language, Date style) Time zone(Country, City)
Price	VAT included	Currency(Country), VAT rate(Country) Currency Exchange Rate (Date issued) Price format(Country, Language)
Physical Quantity	Measure unit, scale Error percentage	Measure System(Country), unit prefix(scale) Quantity format(Country, Language)
Telephone Number		Country calling code(Country) International prefix(Country) Phone format(Country, Language)
Address		Address format(Country, Language)

4 Semantic Annotation of Web Contents

Representing a *CSC* as semantic object requires annotating it with metadata (i.e., a concept S and a set of context attributes C). This section initially gives an insight on the different annotation techniques, namely external and internal annotation. Then, it details the advantages of internal annotation and RDFa technology for the purpose of this paper.

4.1 Annotation Techniques: External vs. Internal

Document annotation can be external and internal. With external annotation, metadata is represented and stored in an external annotation document. Then, these metadata refers to a part of a document (typically an XHTML tag) that is annotated using a pointing language such as Xlink and Xpointer. For instance, the value V of the aforementioned date semantic object can be annotated using this technique as follows. The concept S and the context information C are represented inside an RDF statement as a set of RDF attribute-value pairs. The

subject of this statement is an Xpointer (e.g., /pageA#Xpointer(html/body/div[2])) which refers to the second *div* element. The latter represents the date value V inside the annotated document (page A)[3]. The main motivations behind external annotation are twofold. First, it provides a way to annotate already-published HTML documents without changing them and to annotate new ones without introducing new elements into their document-type definition (DTD). Second, one annotation document can be reused for annotating specific parts of multiple Web documents. This is useful for annotating (already-published) parts that have common semantics and structures such as calender events and products data [6,7,10].

However, since Xpointers refer to annotated elements based on their paths in the annotated document, this technique requires additional work to synchronize the annotation document with the annotated document. Furthermore, external annotation leads to problems when aggregating contents. First, references to the document structure are modified; and second, aggregation may imply context changes, and as the external annotation files are attached to the original contents, they should not be updated when the context changes due to aggregation in different contexts.

On the contrary, internal annotation stores Web contents and metadata together in the same document. Metadata is embedded as an XHTML attribute of the document element (e.g., XHTML tag) that delimits the Web contents to be annotated. As we will see in the next section, internal annotation remains simple, and the aggregation of annotated contents does not require additional synchronization, since metadata are directly embedded in the document. Then, as long as the document is accessible for editing, both Web content and annotation can be edited, deleted, and/or aggregated without any problems.

However, internal annotation requires additional work for annotating already-published Web documents. Indeed, each XHTML tag that represents a Web content to be annotated should be edited separately to embed the annotation, leading to a redundancy problem. For instance, if two or more date contents are created by one author (may be in the same page), then the same context information needs to be provided in all the corresponding XHTML elements.

To conclude, external annotation faces significant limitations with respect to creation/delete, update, and aggregation of Web contents and metadata. Therefore, we adopt internal annotation, which despite its redundancy drawback, remains the best tradeoff with respect to the Web 2.0 use cases as it eases the creation/delete, update, and aggregation tasks. We rely on the RDFa[4] syntax in order to annotate our documents. We give a short introduction to RDFa in the following before detailing our architecture.

4.2 RDFa-Based Internal Annotation

RDFa provides an annotation syntax to express RDF statements in XHTML documents. It relies on a collection of XHTML attributes such as *about*, *property*,

[3] More details available on W3C annotation note : `http://www.w3.org/TR/annot/`

[4] `http://www.w3.org/TR/xhtml-rdfa-primer/`

and *content* to embed RDF statements in XHTML. Also, it provides processing rules to extract these statements from XHTML [1,3].

This section shows how we utilize RDFa for annotating $CSCs$ as semantic objects and illustrates how it works seamlessly with the Web 2.0 use cases. To keep the paper self-contained, we annotate the date contents presented in Section 2.1 as semantic objects and track the tasks to perform, as shown in Figure 3. We represent the concepts S and the context attributes C as RDF statements and localize them using RDFa syntax.

In page A, the *namespaces* inside HTML tag represent the URLs of RDF constructs and the *XHTML+RDFa* version of XHTML DTD. The RDFa attribute $about = `\#D1'$ identifies a date $SemObj$. The RDFa attribute *property='dc:date'* represents the date concept S and the date contents 07/09/2009 represents the value V of the date after being updated by the American author[5]. In the inner $\langle span \rangle$ tags, the set of RDFa *property* attributes represent the date context attributes, and the RDFa *content* attributes represent the values of the context information related to the American author, as illustrated in Figure 2. Also, the date content *2009-09-10* in page B is annotated with the context information related to the Canadian author in similar way. Next, the annotated dates from pages A and B are aggregated like *"copy and past"* to the page C. It is worth noting that when the American author updates the date contents on page A (i.e., T3), the annotations are updated according to his context too.

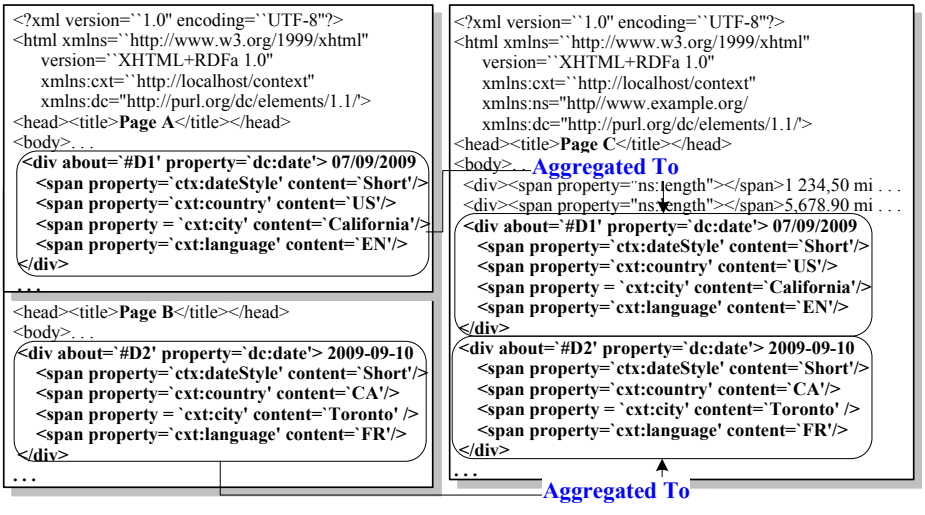

Fig. 3. Internal annotation using RDFa

5 Architecture

In this section, we present our proposed architecture to illustrate how our approach can be deployed and work seamlessly with the existing Web technology

[5] We do not present the original date value created by the British author for brevity.

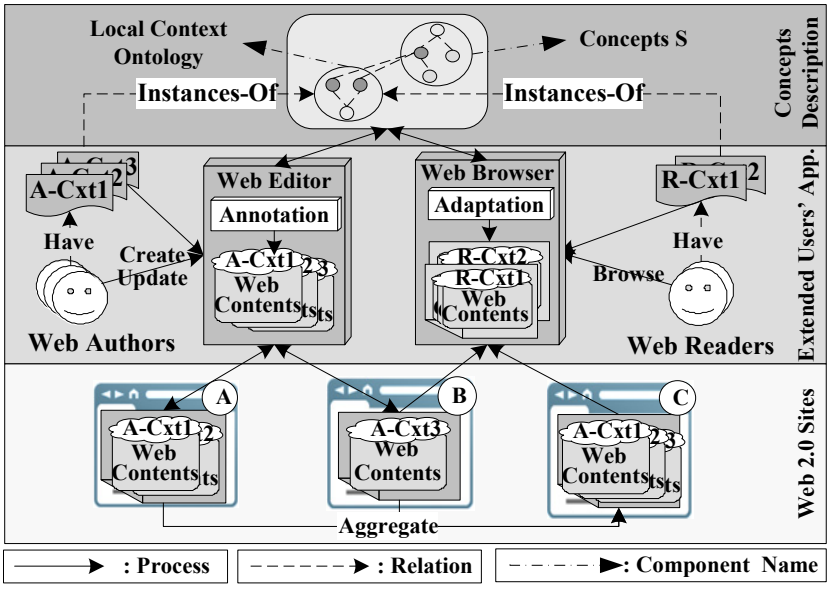

Fig. 4. A general architecture overview

stack. This architecture adds a layer called *concept description* and extends both traditional Web editors and Web browsers with an annotation engine and an adaptation engine respectively, as shown in Figure 4.

The *concept description* layer illustrates the relation between $CSCs$ and context attributes at a conceptual level as described in Section 3. The role of this layer is to provide the necessary vocabularies to specify users' local contexts (i.e., A-Cxts and R-Cxts) and concepts S. We consider that Web authors and readers agreed on these common vocabularies.

The role of the annotation engine is to assist authors for annotating $CSCs$ with their context. Web authors need to specify their context (i.e., A-Cxts) and the type of the CSC to be annotated. Then, the annotation engine interactively annotates the specified CSC with context attributes, which now forms a semantic object $\langle S, V, C \rangle$. Section 6 discusses the annotation process in more details with the extended Web editor. The adaptation engine also allows Web readers to specify their contexts (i.e., R-Cxts). Then, it adapts each annotated CSC from authors' to a reader's context. The output is semantically equivalent to the annotated CSC, but it is represented according to the reader's context. The adaptation process of the extended Web browser are out of this paper scope (See [3] for more details).

6 Annotation Process

This section details our vision on how to interactively accomplish the annotation process. Basically, this process illustrates the role of the aforementioned

annotation engine during a Web author/extended Web editor interaction (Figure 5). Our annotation process consists of one pre-annotation task (i.e., Task 1) and four annotation tasks as follows:

1. **Local context specification**
   Input 1: Author's context attributes C
   Output 1: A-Cxt document
   In this task, the Web author needs to specify his local context. Here, the author must specify static attributes, and one or more dynamic attributes if there is a specific need for unusual/specific dynamic values. Other dynamic attributes are inferred from static context attributes as described in Table 2. The specified values are then stored in the A-Cxt document. This task (pre-annotation task) is thus performed prior to content annotation.
2. **Context attributes extraction**
   Input 2: A concept S, a content V, Context ontology, A-Cxt document
   Output 2: S, V, C
   During content creation or update, the author needs to select (i.e., highlight) the target value V of CSC to be annotated, and then selects a concept S. Upon selection of S and V, this process extracts the corresponding context attributes from the context ontology based on the concept S and its relations with these attributes (See Table 2). After that, the value of static and specified dynamic context attributes are extracted from the A-Cxt document. S, V, and C are utilized as inputs to Task 3.
3. **Annotation creation**
   Input 3: *S, V, C*
   Output 3: $SemObj = \langle S, V, C \rangle$
   Using the inputs received from the Task 2, this process annotates V as a $SemObj$ as follows. First, it builds a semantic object instance $SemObj$ from S, V, and C. Second, it generates the XHTML+RDFa representation of this $SemObj$. Finally, in the extended Web editor interface, it replaces the value V with the generated $SemObj$. Note that, an author can repeat this task and the previous one in order to annotate other $CSCs$.
4. **Annotation testing**
   Input 4: $SemObjs$ in the editing interface, A-Cxt document
   Output 4: Tested $SemObjs$
   During authoring and annotation process, this task scans Web contents in the background and checks the generated SemObjs. To this end, a semantic object instance called $testerSemObj$ is built for each generated $SemObj$. The $testerSemObj$ takes the concept S and the context C of the generated $SemObj$ as parameters and generates a test value TV. Then, the generated $SemObj$ is compared with the tester $SemObj$ as follows. First, the value V is compared with the value TV and provides a warning message if the former does not comply with TV (like a smart tag in Microsoft Word). For example, if an author, by mistake, annotates a length CSC with a date concept, then the generated $SemObj$ is highlighted and provides a warning message (e.g., this content is not a date). Second, it compares V with the context attributes

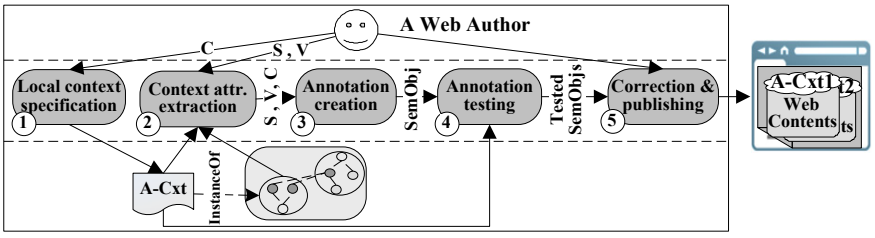

Fig. 5. Details of the annotation process

stored in the A-Cxt. For example, comparing a date content with a date style attribute and provides the corresponding warning message if V violates this attribute.

5. **Correction and publishing**
   Input 5: Tested $SemObj$
   Output 5: Annotated $CSCs$ in a Web page
   In this task, an author needs to correct the highlighted $SemObjs$. For example, he needs to correct the annotation of the length CSC, in the above example, with the length concept (instead of the date concept). Finally, the author publishes the annotated Web contents.

The above tasks provide the means to annotate $CSCs$ in an interactive and easy manner. First, context attributes that are specified in the A-Cxt document can be reused to annotate Web contents at different authoring times. Second, Task 2 enables authors annotating $CSCs$ as easy as formatting text in word processors [12]. Furthermore, authors do not need to know the relations between the $CSCs$ and the local context attributes, since the context attributes and their values are extracted automatically based on the concept S. At the same time our approach is flexible as advanced authors still have the possibility to override dynamic inferred attributes in the annotation. Third, creation of semantic annotation, in Task 3, hides the technical complexities of the RDFa syntax. In addition, it reduces the annotation efforts and the number of annotation errors that could be performed by Web authors. Finally, testing semantic objects, in Task 4, assists authors into correctly authoring $CSCs$ and therefore reduces the potential errors that they could perform.

The annotation process can be extended in one of the following directions. First, it can be extended with information extraction, together with semantic-aware auto-complete interface [8]. Information extraction is used to match typed $CSCs$, at authoring time, with one of the concepts S based on predefined concept patterns for example. If the matching task succeeds, then the semantic auto-complete interface will recommend this concept to the author. This increases the willingness of authors for annotating $CSCs$. Also, it helps authors knowing which $CSCs$ need to be annotated. Second, providers (i.e., Web sites designers or administrators) can relate concepts S and context attributes C to *annotation templates*, such as event and product templates. Based on an

author's context, these templates are generated upon an author's request, and the typed (filled) *CSCs* are annotated accordingly. This is useful for annotating structured contents [11]. Third, the annotation testing task can reuse the information extraction technique to check if there is a *CSC* that matches one of the concept S. If so, then it provides a warning message to the author in order to confirm or deny this matching. This enhances the annotation results, since authors could forget annotating some *CSCs*.

7 Prototype

This section presents a prototype as a proof-of-concept of our approach. Basically, the proposed prototype demonstrates the annotation process presented above. To this end, we use an HTML form to allow authors specify and store their contexts information into A-Cxts. The latter and the context ontology are implemented using RDF/XML syntax, and existing concepts from published ontology (e.g., Dublin Core) are reused as concepts S. We also extends the TinyMCE™WYSIWYG editor[6] with a concept menu. The latter allows authors to select concepts S for annotating *CSCs* typed in the editor. As a back end, we use Java™APIs to extract context attributes and their values for each concept S. Finally, Javascript is utilized to annotate the value V with the selected concept S and the extracted context attributes C as semantic object. Figure 6 presents a screenshot of our prototype that illustrates how the date content from our scenario during Task T3 can be annotated as semantic object.

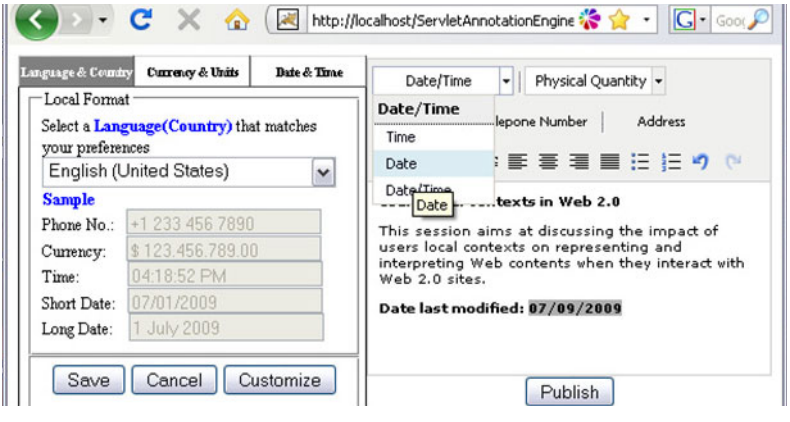

Fig. 6. A screenshot of the extended Web editor

8 Related Work

This section discusses existing works related to Web annotation and users' local contexts. The term *Web annotation* has been used to refer to a process of

[6] `http://tinymce.moxiecode.com/`

adding metadata to a Web document (e.g., XHTML document) or to metadata itself [6,10]. In practice, annotation is performed manually by authors or automatically by software application. Manual annotation incurs several problems such as usability, error proneness, scalability, and time consumption. Automatic annotation usually relies on information extraction techniques and/or machine learning to identify and annotate Web contents based on a natural language knowledge base (e.g., WordNet). However, automatic annotation faces concept disambiguation problems. Indeed, many Web contents have overlapping semantics and an annotation application cannot correctly recognize them all [5,15].

Interactive annotations have been commonly used recently [5]. Amaya is a W3C annotation framework that enables authors to add and share notes about Web contents. These notes are created and updated by authors, represented using RDF schema, associated with contents using XPointer, and can be stored on authors' machines or on a remote annotation server [10]. Saha [15] is another interactive annotation system. It aims at enabling authors to enrich Web documents with ontological metadata and uses the latter as a semantic index. Saha uses a set of predefined ontological schemas and associates them with annotated documents using Xpointer.

In the domain of the Web 2.0, several systems have relied on interactive annotation and on RDFa. For instance, Luczak-Roesch and Heese [12] propose a system that enables authors to annotate Web contents and publish them as RDF-based linked data using RDFa. Also, semantic wikis such as SweetWiki[7] and OntoWiki[8] allow authors to annotate wiki contents with RDF-based predefined metadata (e.g., FOAF) using RDFa.

Local context has been acknowledged as an important issue [4,13,14]. However, a few approaches have used annotations for handling users' contexts. Furthermore, most existing approaches rely on two assumptions. First, Web contents in a single Web page are represented according to one context only. Second, context information such as users' preferences or users' device capabilities are acquired into a context model, and then different contents are provided to different users. For example, transcoding systems annotate Web contents with transcoding metadata to define contents' roles. Then, annotated contents are adapted (e.g., restructured, summarized, or deleted) based on users' device capabilities or based on special requirements of users (i.e., visually impaired) [7].

To conclude, to our best knowledge, none of the existing approaches enable authors to annotate Web contents with their local contexts, so that Web browsers can handle local contexts' discrepancies. We advocate that our approach is one step further towards enriching the semantics of Web contents. Several works can be extended with this approach.

9 Conclusion

Today, the Web has evolved to a new era characterized by authoring and sharing of Web contents via different Web users and sites, known as Web 2.0. This

[7] `http://semanticweb.org/wiki/SweetWiki`
[8] `http://ontowiki.net/`

evolution leads to misunderstandings of Web contents as users from different communities use their own local contexts to represent and interpret these contents. This paper proposes a context-aware interaction approach that enables authors enriching Web contents with context information as easily as formatting text. Accordingly, it becomes feasible for Web browsers to personalize annotated contents according to different users' contexts. We present an architecture and a prototype to show how our approach works seamlessly with the Web technology stack. As a future work, we plan to set up an experiment to evaluate the practical feasibility of our approach from both authors' and readers' perspectives.

References

1. Adida, B.: hGRDDL: Bridging microformats and RDFa. J. Web Sem. 6(1) (2008)
2. Al-Jabari, M., Mrissa, M., Thiran, P.: Handling users local contexts in web 2.0: Use cases and challenges. In: AP WEB 2.0 International Workshop. CEUR Workshop Proceedings, vol. 485, pp. 11–20 (2009)
3. Al-Jabari, M., Mrissa, M., Thiran, P.: Towards web usability: Providing web contents according to the readers contexts. In: Houben, G.-J., McCalla, G., Pianesi, F., Zancanaro, M. (eds.) UMAP 2009. LNCS, vol. 5535, pp. 467–473. Springer, Heidelberg (2009)
4. Barber, W., Badre, A.: Culturability: The merging of culture and usability. In: The 4th Conference on Human Factors and the Web (1998)
5. Corcho, Ó.: Ontology based document annotation: trends and open research problems. IJMSO 1(1), 47–57 (2006)
6. Handschuh, S., Staab, S.: Authoring and annotation of web pages in CREAM. In: WWW, pp. 462–473 (2002)
7. Hori, M., Kondoh, G., Ono, K., Hirose, S., Singhal, S.K.: Annotation-based web content transcoding. Computer Networks 33(1-6), 197–211 (2000)
8. Hyvönen, E., Mäkelä, E.: Semantic autocompletion. In: Mizoguchi, R., Shi, Z.-Z., Giunchiglia, F. (eds.) ASWC 2006. LNCS, vol. 4185, pp. 739–751. Springer, Heidelberg (2006)
9. Ishida, R.: Making the World Wide Web world wide. W3C internationalization activity, `http://www.w3.org/International/articlelist/` (last accessed: February 19, 2010)
10. Kahan, J., Koivunen, M.-R., Hommeaux, E.P., Swick, R.R.: Annotea: an open RDF infrastructure for shared web annotations. Computer Networks 39(5) (2002)
11. Kettler, B.P., Starz, J., Miller, W., Haglich, P.: A template-based markup tool for semantic web content. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 446–460. Springer, Heidelberg (2005)
12. Luczak-Roesch, R.H.M.: Linked data authoring for non-experts. In: Proceedings of the WWW '09, Workshop Linked Data on the Web, LDOW 2009 (2009)
13. Reinecke, K., Bernstein, A.: Culturally adaptive software: Moving beyond internationalization. In: Aykin, N. (ed.) HCII 2007. LNCS, vol. 4560, pp. 201–210. Springer, Heidelberg (2007)
14. Troyer, O.D., Casteleyn, S.: Designing localized web sites. In: Zhou, X., Su, S., Papazoglou, M.P., Orlowska, M.E., Jeffery, K. (eds.) WISE 2004. LNCS, vol. 3306, pp. 547–558. Springer, Heidelberg (2004)
15. Valkeapää, O., Alm, O., Hyvönen, E.: An adaptable framework for ontology-based content creation on the semantic web. J. UCS 13(12), 1835 (2007)