ABSTRACT

Social tagging is one of the hallmarks of Web2.0. The most common role of tags is descriptive. However, tags are being used for other purposes such as to indicate some actions to be conducted on the resource (e.g., “toread”). This work focuses on “prescriptive tags” that have associated some implicit behaviour in the user’s mind. So far, little support is given for the automation of this “implicit behaviour”, more to the point, if this behaviour is outside the tagging site. This paper introduces the notion of “reactive tags” as a means for tagging to impact sites other than the tagging site itself. The operational semantics of reactive tags is defined through event-condition-action rules. Events are the action of tagging. Conditions check for additional data. Finally, rule’s actions might impact someone else’s account in a different website. The specification of this behaviour semantics is hidden through a graphical interface that permits users with no programming background to easily associate “reactions” to the act of tagging. A working system, TABASCO, is presented as proof of concept.

ACM Classification Keywords
H.5.3. Group and Organizational Interfaces: Computer supported cooperative work

General Terms
Experimentation

Authors Keywords
Tagging, ECA rules, interoperability, SIOC, RULE-ML

1. INTRODUCTION

Tagging is an important task for different systems and services such as Delicious\(^1\), WordPress\(^2\) or Flickr\(^3\) which allow participants to annotate a particular resource (e.g. a web page, a blog post, an image) with a freely chosen set of keywords (a.k.a tags). Tags can be a powerful tool for social navigation [14], helping people to share and discover new information contributed by other community members. Notice however that such collaboration is restricted to the site itself. Collaboration-wise, these websites behave as islands where collaboration is restricted to resources and users within the website walls.

However, it is very common for users to keep an account in distinct tagging sites depending on a broad range of issues: the resource type (e.g. if bookmarks then, Delicious; if video then, YouTube\(^5\)), the supporting community (e.g. if music-related resources such as mp3, videos, lyrics are the resources to tag, Last.fm\(^6\) could be an appropriate site), confidentiality (e.g. if restrict sharing is an issue, you might favour to use www.bookmarks2.com rather than Delicious where private bookmarks are cumbersome to handle), etc. Therefore, taggable resources will most likely be scattered throughout the Web. The potential synergies among many sites, communities, and services are expensive to exploit, and their data are difficult and cumbersome to link and reuse. The main reason for this lack of interoperability is that for the most part in the Social Web, common standards still do not exist for knowledge and information exchange and interoperation. This paper addresses communication in such a setting.

This paper introduces a framework for Tag-BASEd, intersite Communication (TABASCO). The system permits users to communicate seamlessly through heterogeneous websites. Users are represented through their website accounts. Tasks are those set by the websites themselves, and normally available through an API. Tags are the means to denote the message that encrypts the associated task in the target account (hereafter referred to as “reactive tags”). Messages are originated in the sender website and impact on the receiver website. Finally, web resources (e.g. bookmarks, blog posts, etc) stand for message parameters.

As an example, let U1 and U2 be two users that hold an account in Delicious and RememberTheMilk\(^7\) (RTM), respectively. Delicious keeps bookmarks, and supports tagging. RTM manages to-do lists, and permits to file and

\(^{1}\)http://www.delicious.com/
\(^{2}\)http://wordpress.org/
\(^{3}\)http://www.flickr.com/
\(^{5}\)http://www.youtube.com/
\(^{6}\)http://www.lastfm.es/
\(^{7}\)http://www.rememberthemilk.com/
prioritize tasks. In this example, Delicious and RTM will play the role of the sender and receiver sites, respectively. U1 wants to communicate to U2 when a given bookmark stands for a paper worth reading. To this end, U1 tags the worth reading bookmark in Delicious as "toread:1" where "toread" stands for a priority parameter. This tag is a reactive tag, i.e. its reactive semantic has been previously defined in TABASCO by U1 provided he holds U2 authorization. This makes TABASCO monitor U1 tagging behaviour in Delicious. When "toread" is used, TABASCO enacts its associated semantics: adding a new task in U2’s RTM account. Both the task’s title and priority are obtained from the bookmark and the tag parameter, respectively.

TABASCO aims at binding disperse communities together. From a communication perspective, the approach accounts for uniformity and site independence. So far, communication is provided within the site’s boundaries through distinct mechanisms: button, command lines or even tags (e.g. the so-called machine tags in Delicious, e.g. for:zon). TABASCO uses reactive tags for messaging along no matter the website. Any website supporting tagging is liable to use reactive tags. From the website perspective, reactive tags are just standard tags. It is TABASCO monitoring what makes the tag be reactive. Reactive semantics is specified using Event-Condition-Action rules. Site interoperability is achieved using ontologies. Specifically, Friend of a Friend (FOAF)\(^8\), Semantically-Interlinked Online Communities (SIOC)\(^9\) and ECA-ML [13] ontologies are used.

TABASCO does not require any plugin on participating sites. Sender sites need to provide tagging capabilities. Receiver sites should support an API to programmatically interact with the site (e.g. adding a resource). The specifics of both senders and receivers are encapsulated using a driver-like mechanism. Once the driver is on place, TABASCO can monitor the tagging behaviour (for sender sites), and enact the desired functionality (for receiver sites). The same site can play both roles: sender and receiver. So far, reactive tags only operate on creating new entries on tagging sites.

Our research contributions are twofold. First, we propose a new way to classify tags based on their behavioural semantics. We introduce “reactive tags” and illustrate the use of such tags for cross-site collaboration. Second, we offer a knowledge model (a way to describe) and an execution model (a way to enact) reactive tags. TABASCO is presented as proof of concept.

### 2. TAG CLASSIFICATIONS: A NEW PERSPECTIVE

Tags capture a broad variety of semantics. Several studies address this heterogeneity from different perspectives. Ames et al. [2] expose a classification based on a user motivation to tag. Goldar et al. [9] classify tags based on the semantic of a tag, proposing two set of tags: descriptive tags, which denote resource-related information (type, what_abouth, place, etc); and personal tags, which capture subjective information (affective, impressions, tasks, etc). Recent studies [11, 16] have demonstrated that an increasing number of tags are related to personal annotations that describe user tasks (e.g. toread), subjective (e.g. important) or affective (e.g. cool) emotions. Some of these tags usually conveys some associated user behaviour when annotate some resources with these tags (e.g. toread). However, this behaviour resides in the user’s mind. We focus on these tags.

To better frame our work, Table 1 introduces two dimensions for tag classification: intend subject versus intend enactor. Tag intend can be descriptive or prescriptive. This intention is conducted by an “intend enactor”: an end user (i.e. the tagger) or a machine (i.e. the tagging website). Next paragraphs present the so-identified four quadrants.

<table>
<thead>
<tr>
<th>User + Descriptive: Standard Tags.</th>
</tr>
</thead>
<tbody>
<tr>
<td>So far, most tagging sites focus on this kind of tags. Although they can serve a broad range of purposes [9], no support is provided from the tagging site, except search and classification. All tags are handled in the very same way no matter their intend. The tag is for user consumption: from users to users.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Machine + Descriptive: Machine Tags.</th>
</tr>
</thead>
<tbody>
<tr>
<td>These tags describe the resource in a machine-readable format. The description can include the resource format, upload date, geographical coordinates, etc. So-called machine tags(^10) can be provided by either the user himself or the tagging site. Flickr illustrates the first case. A special syntax is provided for users to input machine tags. Specifically, machine tags are described through a triplet: (&lt;namespace,property,value&gt;) (e.g. geo:latitude=... , flickr:user=...). Additionally, machine tags can be automatically introduced by the website itself. For instance, Delicious automatically creates machine tags to make resources easier to find such as system:media:audio or system:filetype:mp3. Amazon(^11) also provides machine tags to describe its books (e.g. amazon:asin=1234566). Regardless of whether the tag provider is a human being or a machine, the final addressee of the tag is a machine.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Machine+Prescriptive: Action Tags.</th>
</tr>
</thead>
<tbody>
<tr>
<td>While descriptive tags are passive, prescriptive tags convey some associated behaviour. If this behaviour is set by the website then, the term “action tags” is used. For instance, action tags are available in Delicious and Flickr. Delicious offers a “for:User” action tag. Tagging a bookmark with “for:U2” causes Delicious to automatically send a notification with this bookmark to U2’s account. Flickr offers through its Zonetag(^12) the “blog:BlogName” action tag. In this case, the associated action is to upload the photo to the specified blog.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 1: Tag classification: intend subject versus intend enactor.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>Machine</td>
</tr>
<tr>
<td>User</td>
</tr>
</tbody>
</table>

\(^{8}\)http://xmlns.com/foaf/spec/  
\(^{9}\)http://rdfs.org/sioc/spec/  
\(^{10}\)http://www.flickr.com/groups/api/discuss/72157594497877875  
\(^{11}\)http://www.amazon.com/  
\(^{12}\)http://zonetag.research.yahoo.com/
sent to U2, i.e. tagged as ‘for:U2’). This associated behavior is enacted at tagging time. Besides the resource being tagged, additional parameters can be provided using the ‘:*’ notation.

**User+Prescriptive: Reactive Tags.**
Most surveys identify one kind of tag to annotate the user’s intentions on the resource being tagged [2, 4, 9]. Common examples include “toDo,” “toRead”, “toDownload”, etc. They are prescriptive tags but provided by users. Unfortunately, the prescription as such rests on the users’ mind. So far, no way exists for the user to define and automate this behavior. Additionally, this behavior tends to surpass the tagging site to impact other places (e.g. the printer, your to-do list, etc), and might affect other users (e.g. the aforementioned for:U2 example where U2 is notified of a U1 action). This work is an attempt for a site-independent, communication-centric approach to the definition and enactment of reactive tags. Next section introduces the requirements for such a system through two case studies.

### 3. ANALYSIS & SCENARIOS

For the purpose of this work, we perceive a Community as a subset of the Cartesian product of \( U \times W \), where \( U \) stands for the users of the community, and \( W \) denotes the set of dedicated websites. Basically, a community denotes the set of accounts that users \( U \) hold in sites \( W \). Additionally, a Collaboration Space captures the set of messages that can be eventually exchanged among community members. More formally, a Collaboration Space is a directed, edge-labelled graph where nodes stand for user accounts of the community, and edges denote potential messages to be exchanged among user accounts. The edge labels stand for the reactive tags.

As an example, consider users Jon, Oscar and Iker to conform a research group. Dedicated websites are used to support their work: Delicious to store bookmarks, WordPress to publish research notes, and RTM to to-do lists. Their accounts conform the sample community:

\[
sampleCommunity = ((oscar, Delicious), (oscar, WordPress), (oscar, RTM), (jon, Delicious), (jon, RTM), (iker, Delicious), (iker, WordPress))
\]

This sampleCommunity provides the grounds to define the Collaboration Space. User accounts become nodes whereas edges define collaboration relationships between user accounts. An edge is drawn from a source account to a target account. An edge has a label (i.e. the “reactive tag”), and an operational semantics that describes how actions on the source account impact the target account. Next paragraphs introduce two running scenarios.

**Scenario 1.** Oscar tags with “toread” interesting papers in his Delicious accounts. Although the list of papers to read could be obtained from Delicious by searching for the “toread” tag, Delicious is not a scheduling tool. Hence, Oscar keeps a “myReadingList” folder in RTM. RTM provides means to monitor, schedule or share to-do tasks. However, this implies a duplication between Delicious and RTM. To keep both lists in syncrony and avoid manually recording data twice, Oscar decides to make “toread” reactive, i.e. associating an operational semantics that indicates when synchronization occurs (at the time of tagging), and how it occurs (moving some data from Delicious to RTM).

Specifically, the source node and the target node are (oscar, Delicious) and (oscar, RTM), respectively. The operational semantics can be read as follows: “on tagging toread at Oscar’s Delicious, do create a task in the myReadingList folder on Oscar’s RTM”. Finally, RTM permits to assign priorities to tasks. This priority can not be derived from the bookmark but it can be provided as a tag parameter (e.g. “toread:3”).

**Scenario 2.** Oscar is in charge of browsing through the latest journal issues. On detecting an interesting paper, he writes a review on his WordPress blog, and next, sends this post’s URL (a.k.a. permalink) to the inbox of Jon’s Delicious. Since this is done in a routine basis, Oscar defines a reactive tag: “review”. The source node is (oscar, WordPress). The target node is (jon, Delicious). The operational semantic is “on tagging review at Oscar’s WordPress, bookmark this WordPress post at Jon’s Delicious”. However, this service can be extended to other members of the community. That is, the target node can stand for several accounts. Account aggregates are based on FOAF relationships (see section 5.2). We can then use these standardized relationships to describe members intentionally through their relationships. For instance, using FOAF’s work with relationship, we can introduce the target node (work with, Delicious) to represent the Delicious accounts of those mates working with Oscar (i.e. the source node). Besides simplicity, this approach accounts for evolvability: new colleagues of Oscar readily enjoy this service as the FOAF file is updated.

Previous scenarios illustrate the notion of Collaboration Space as a graph of nodes (i.e. user accounts), and labelled edges (i.e. reactive tags). Edges introduce collaboration paths whereby tagging on the source node triggers some site-dependent reaction on the target node. The following key requirements are identified:

- **usability.** TABASCO is targeted to end users. It should be designed to be extremely lightweight to use, and intuitive to grasp. Users can define reactive tags at any time, in the very same way that traditional tags are introduced as the necessity arises,

- **interoperability.** TABASCO provides an additional layer on existing tagging systems. This brings issues on both syntactic interoperability (e.g. data formats, communication protocols) and semantic interoperability (e.g. existence of a shared reference model),

- **integrity.** Integrity is the assurance that information can only be accessed or modified by those authorized to do so. TABASCO extends tagging consequences outside a single user account. Tagging on one user account might impact someone else’s user account. Users should keep control of who and how are their accounts accessed.

### 4. TABASCO AT WORK

A TABASCO installation supports multiple Collaboration Spaces\(^1\). Creating a Collaboration Space goes along four main steps. This section outlines the main TABASCO GUI for this purpose.

\(^1\)An on-line version is available on http://www.owkin.org/tabasco
user authorization token granted by hold the username and password of the account but just an token" to TABASCO to work on Delicious out tokens. A token grants access to a specific site (e.g. OAuth, WordPress, RTM). This is accomplished by using OAuth. This process is internal to TABASCO, and it does not involve any additional interaction with the website (e.g. Delicious).

Registration ("My Account" tab: Figure 1(a)). Users first indicate whether their accounts will become nodes of the Collaboration Space. So far, accounts are limited to Delicious, WordPress and RTM. The process goes as follows: (1) the user selects the website (e.g. Delicious), (2) TABASCO asks Delicious to authenticate the current user, (3) the user is re-directed to Delicious where the user identifies himself, and if granted, (4) Delicious will provide “an authorization token” to TABASCO to work on this user account.

It is most important to note that TABASCO does not hold the username and password of the account but just an authorization token granted by Delicious on behalf of the user. This is accomplished by using OAuth (Open Authorization). OAuth is an open standard for authorization that allows users to share their private resources (e.g. bookmarks) without having to hand out their credentials, typically username and password. This is achieved by handing out tokens. A token grants access to a specific site (e.g. RTM) for specific resources (e.g. tasks at myReadingList folder) and for a defined duration (e.g. the next 2 months).

Authorization request ("My Community" tab: Figure 1(b)). Even if TABASCO holds an authorization, this does not imply that any registered user can enjoy this authorization. Rather, defining reactive tags over a user account requires authorization privileges upon this account. The petition lifecycle goes along the following stages: start, pending, accepted/rejected and revoked.

Authorization grantee ("My Grantees" tab: Figure 1(c)). Authorization petitions are managed by account owner themselves. Petitions are notified through the “mail” icon, and handled through the “My Grantees” tab. If granted, TABASCO extends the credential to the petitioner so that he can now define reactive tags on this account. Authorization can be revoked at any moment by the account owner. This process is internal to TABASCO, and it does not involve any additional interaction with the website (e.g. Delicious).

Reactive Tag Definition ("My Reaction Tags" tab: Figure 2). Tag definition includes (1) the source node, (2) the target node, (3) the label and (4), the operational semantics. In Figure 2, the left-hand side panel provides available nodes according to the authorizations held by the current user. Source nodes are restricted to accounts owned by the user. That is, a user cannot define a reactive tag that departs from someone else’s account. Target nodes correspond to accounts the user is authorized to operate upon. This includes his own accounts plus those he has been granted authorization. Through drag&drop, the user initializes the middle canvas with the desired nodes. Standing for user accounts, nodes are depicted as a blend of the user picture and the website icon. Edges can now be drawn between user accounts, and in so doing, setting the operational semantics of tags.

The operational semantics describes how a reactive tag is interpreted as sequences of computational steps. These sequences then are the meaning of the tag. As an attempt to find a compromise between expressiveness and usability, this operational semantics is restricted to be “transformational”. That is, websites are regarded as silos of items (bookmarks for Delicious, tasks for RTM, posts for WordPress). The semantics indicates how an item of the target site can be obtained from an item of the source site.

As an example, consider our first scenario: the semantics of the toread tag (see Figure 2): “on tagging toread at Oscar’s Delicious, do create a task in the myReadingList folder on Oscar’s RTM”. The type of both items (i.e. Bookmark and Vtodo) is set by the participating websites (i.e. Delicious handles Bookmarks while RTM manages Vtodo items). The type of these items (i.e. their properties and structure) are being described along the SIOC ontology initiative (see section 5).

Since, item types are known in advance, TABASCO already sets partial transformation for all possible combinations. For our sample case, three types of items are involved: BlogPost (for WordPress), Bookmark (for Delicious) and Vtodo (for RTM). This implies nine possible transformations: BlogPost to BlogPost, BlogPost to Bookmark, BlogPost to Vtodo, etc. It could be possible to define a canonical model that factors out the n item types so that the number of combinations would be reduced from \( n \times n \) to \( 2 \times n \). However, the over-
neered in TABASCO. From this perspective, edges are envisioned as pipes that push items along the Collaboration Space.

However, these transformations are not always complete. That is, some properties of the target item might not be obtained from the source item. For instance, Vtode properties include "categories" and "priority". These properties cannot be obtained from a Bookmark. However, two other source of data are possible (see the pop-up form in Figure 2). First, the semantics of the tag can provide some properties as constants. For instance, "toread" implies the so-created tasks to be located at the myReadingList folder (i.e. categories = "myReadingList"). Another option is for properties to be set at tagging time. "Priority" is a case in point. Now, "priority" becomes a parameter of the reactive tag (e.g. "toread:priority"), and hence, it is provided by the user when tagging. For efficiency sake, this option requires to know in advance the range of values this parameter can take\(^{17}\) (e.g. 1, 2, 3). This can be a limitation if such values can not been foreseen. So far, a single parameter has been enough to tackle the scenarios considered. It is technically feasible to introduce additional parameters, should this be required.

To conclude, reactive tags are envisaged as functions (i.e. \(\text{tagname}(\text{inputItem}) \rightarrow \text{outputItem}\)) where tag semantics is specified as a transformation between website resources. Parameters of the target resource can be obtained from (1) the source item, (2) a tag's constant, or (3) a tag's parameter provided by the user at tagging time. Such simplicity permits the semantics to be graphically specified and hence, to be easily provided by the end user himself (see Figure 2). Finally, clicking on the "draft" button provides the code that realizes such operational semantics (see Figure 3). And this moves us to the implementation.

5. IMPLEMENTATION

A Collaboration Space is a directed, edge-labelled graph where nodes stand for user accounts of the community, and edges denote potential messages to be exchanged among these user accounts. Implementation-wise, these user accounts are envisioned as both producers of events when certain situation arises (i.e. a bookmark is created with a certain tag), and consumers of actions (i.e. create a task). The intertwine of events and actions is orchestrated through reactive tags. Reactive tags are supported as Event-Condition-Action (ECA) rules: when an event (a happening of interest) occurs, evaluate a condition (possibly after querying for some extra data), and if the condition is satisfied then, execute an action. This approach is widely available in databases where ECA rules are known as triggers [15].

Unlike databases, our setting is distributed: events raised in Delicious can cause an action to be performed in RTM. Delicious and RTM are autonomous sites. Being autonomous sites brings not only operational challenges but also semantic ones as the way to describe resources can diverge among sites. This implies to abstract from working sites into higher level descriptions by using ontologies. This is a must for websites "to collaborate" in an heterogeneous and open environment.

Figure 4 introduces the different concerns (i.e. websites, user accounts, rules) and their ontology counterparts. Such mixture is one of the Semantic Web best practices: reusing existing ontologies leads to better data interoperability. TABASCO, an engine to support Collaboration Spaces, is then based on an intensive use of ontologies. Hence, we first introduce the underlying ontologies and next, the architecture.
Figure 3: Reactive tags as ECA rules: the “toread:{$PRIORITY}” case.

5.1 Website Specification
Tagging sites are heterogeneous. Means are needed to abstract away from the specifics of each tagging site, and come up with a common model. For this purpose, we use the SIOC ontology (Semantically Interlinked Online Communities ontology). The SIOC initiative aims to enable the integration of online-community information by providing a common ontology that can hide the heterogeneity of Social Web sites. In our case, we focus on a special case of Social Web sites: tagging sites. SIOC has then been adopted and extended to cater for tagging specifics.

According to its proponents, SIOC Core Ontology is created “using the terms for describing Web-based discussion areas such as blogs and message boards. For example, users create posts (sioc:Post) organized in forums (sioc:Forum), which are hosted on sites (sioc:Site). In parallel with the evolution of new kinds of Social Web sites, these concepts became subclasses of higher-level concepts that were added to SIOC as it evolved: data spaces (sioc:Space, a place where data resides), containers (sioc:Container, used for grouping items together), and content items (sioc:Item)” [3] (see Figure 4). In so doing, SIOC permits to classify and structure the resources commonly found on social Web sites.

The HCLS initiative\(^\text{18}\) describes all the SIOC Containers and proposes a SIOC ontology extension to capture main item types\(^\text{19}\). Specifically, the SIOC Types module defines subclasses of the SIOC Core concepts that we can use to describe the structure and types of content of social websites. We have used this facility to introduce Bookmark as a subclass of sioc:Item.

As an example, WordPress is introduced as a SIOC site in Figure 5 (lines 8-12). A site is described through a title, a description and a logo. WordPress is abstracted as a container of sioc:BlogPost items. Figure 6 provides an example of one such item (namespaces are omitted).

5.2 User-account Specification
User accounts denote associations between websites and users. Previous paragraphs address website description along SIOC ontology. Now, we introduce user description along the FOAF ontology. This ontology is extended with the Relationship\(^\text{20}\) ontology to describe relationships between persons. Figure 5 shows the FOAF description for our sample case. Specifically, the "works-with" association indicates that Oscar works with Jon and Iker.

Figure 4: Ontologies in used at TABASCO.
task in the myReadingList folder on Oscar’s RTM with priority \{SPRIORITY\}.

Rule’s event (lines 12-25). The rule is to be triggered when Bookmarks are created. The event is risen when enacting a sioca:Action that creates a Bookmark at the user account hold by the sioc:has_creator property (i.e. http://www.delicious.com/oscaronekin). Notice an implicit condition that requires the sioca:Action to create an bookmark:Bookmark whose bookmark:hasTopic matches \{toread \{SPRIORITY\}\}. That is, the bookmark must be tagged this way for the rule to be fired. At execution time, the event payload is bound to the OCCURRENCE variable.

Rule’s query (lines 26-47). In this case, the query clause serves to instantiate some auxiliary variables to be later used in the action. Basically, this stands for the built-in mapping parameter (kept in PRIORITY). That is, the bookmark must be tagged this way for the rule to be fired. At execution time, the event payload is bound to the OCCURRENCE variable.

5.5 System Architecture

The system architecture revolves around three main components: the rule engine, the SIOC brokers and the SIOC sites. Figure 7 outlines main component interactions for both rule registration (solid lines) and rule enactment (dashed lines).

The Rule Engine. MARS framework is used as the rule engine [1]. Its functions include: registering ECA rules (ECA Engine), signaling event occurrences (Event Engine), and enacting actions (Action Engine). Events and actions should be defined for the domain at hand. Its realization and definition are externalized from MARS into the SIOC brokers.

SIOC brokers. Three brokers were implemented. They are in charge of event detection (SIOC Event Broker), action enactment (SIOC Action Broker), and condition testing (SIOC Condition Broker). As for the Event Broker, only one type of event is so far handled: item created. At rule definition time, the MARS’s Event Engine indicates to the Event Broker the event to be monitored. This includes the user account to be tracked as well as the reactive tag whose semantic this rule supports. Event tracking is achieved using an RSS reader. Implementation-wise, Event Brokers are

Figure 5: Mixing FOAF & SIOC for describing user accounts.

(lines 17, 18). Other associations can be defined along the Relationship ontology.

Once sites and persons are introduced, we can proceed with accounts. Account and credential data is captured through the sioc:UserAccount class. As Figure 4 depicts, “sioc:Item” individuals are created through “sioc:UserAccount” individuals.

Figure 5 provides an example for Oscar account in WordPress (lines 20-26). If the URL is available then, the resulting instance is linked to the real world account. For instance, the URL http://oscaronekin.wordpress.com denotes the real account of Oscar in WordPress. This FOAF file can then be checked by rules’ conditions to retrieve who are the actual users that works with Oscar, and enact the appropriate actions on their user accounts.

5.3 ECA Rule Specification

MARS’ ECA-ML ontology introduces class eca:rule which comprises eca:event, eca:condition (eca:query) and eca:action classes [1]. MARS’ ontology needs to be customized for the domain at hand. That is, what is an event, what state is to be consulted or what actions can be enacted are all domain specific. For our case, events and actions occur as a result of interacting with websites. Websites are abstracted as SIOC site. Operating with SIOC site is generalized in the sioca:Action class as described in [5].

Despite its name, sioca:Action stands for an event class, specifically, the point in time at the end of the namesake action (“after” events in database parlance). The sioca:Action class holds the sioca:creates property that links to the “sioc:Item” being created, and the sioca:has_creator property that holds a sioc:UserAccount individual.

5.4 An Example

Figure 3 provides an example for the tag toread: \{SPRIORITY\}: “on tagging toread at Oscar’s Delicious, do create a
based on an implementation of the PubSubHubbub protocol.\(^{21}\)

Action Brokers execute SIOC actions as commanded by the Action Engine. So far, actions are limited to item creation. Unfortunately, considerable variations exist among the communication protocols, domain schemas, and APIs supported across social websites. To shelter from these variations, the JSBC (Java SIOC site Based Connectivity) API has been developed. This API mimics JDBC\(^{22}\) specifications where "SIOC drivers" are used to encapsulate website heterogeneity. Each SIOC driver encapsulates the peculiarities of the SIOC site at hand (e.g. protocol, data format, etc). Basically, this means to map abstract SIOC actions into their realization in terms of proprietary API calls of the target website.

SIOC sites. Websites are abstracted as SIOC content providers. This abstraction layer is achieved through drivers. So far, drivers are available for Delicious, WordPress and RTM. Additional sites can be included as long as appropriate drivers are provided. This approach accounts for seamlessly adding new SIOC sites by just realizing the JSBC interfaces. If reactive tags are to be used over YouTube then, JSBC interfaces need to be provided for YouTube.

6. REQUIREMENTS REVISED

Interoperability. Smooth system interoperation is achieved by abstracting site specifics through ontologies. User accounts, websites and the semantics of reactive tags are all captured by adapting existing ontologies, namely, FOAF, SIOC and ECA-ML, respectively. Drivers are introduced to map site specifics into ontology terms, and vice versa. Our experience so far is that driver construction is a five-week effort on average.

Integrity. TABASCO behaves as a “guarantor” of OAuth tokens. Users grant/request authorization tokens to/from TABASCO. This requires the previous consent from the authorization owner. At any moment, owners can check the status of their tokens, and revoke authorizations. This disables the affected rules (i.e. affected tags are not longer reactive) but does not delete them. If the authorization is later renewed, these rules are enabled again. Rule deletion should be explicitly conducted by the creator. Rule deletion does not imply the removal of the companion reactive tag from the source site. So far, authorizations are limited to consult (i.e. read) and create (i.e. write) resources. Finer granularity can be envisaged based on the API operations provided by social sites. Unfortunately, the lack of standards hinders this attempt.

Usability. Along ISO definition [7], we evaluate usability along three aspects:

- Effectiveness, which is the accuracy and completeness with which users achieve certain goals. Indicators of effectiveness include quality of solution and error rates. The “quality of solution” is used as the primary indicator of effectiveness, i.e. a measure of the outcome of the user’s interaction with the system.
- Efficiency, which is the relation between (1) the accuracy and completeness with which users achieve certain goals and (2) the resources expended in achieving them. Indicators of efficiency include task completion time and learning time. In this study, we use “task completion time” as the primary indicator of efficiency.
- Satisfaction, which is the users’ comfort with and positive attitudes toward the use of the system. Users’ satisfaction can be measured by attitude rating scales such as SUMI [12]. In this study, we use “preference” as the primary indicator of satisfaction.

The experiment was conducted among 10 PhD students. All of them had accounts in social websites (they knew about Delicious, WordPress and RTM) but they were not familiarized with the notions of “reactive tag”. Hence, a one-hour talk was given introducing the purpose and functionality of TABASCO.

For the experiment, an account in Delicious, WordPress and RTM was created for each student. The task was to create reactive tags for the two sample scenarios in Section 3. We measure abilities for registering, reactive tag definition and authorization handling\(^{23}\). The results follow.

- Effectiveness. Nine students complete their tasks without any additional help. One student had problems to correctly distinguish between source accounts and target accounts. This makes us think that the GUI is intuitive enough. One issue was the cluttering of the rule definition screen (see Figure 2). This screen is used not only for rule definition but also for rule consultation. When on consultation mode, this screen displays those rules that affect the current user. This includes his own rules plus those defined by other users that act upon his accounts. So far, all such rules are readily depicted. This can lead to cluttered screens. A solution could be the use of the left panel as a kind of filter. This panel displays source/target accounts.

\(^{21}\)http://code.google.com/p/pubsubhubbub/

\(^{22}\)http://java.sun.com/javase/6/docs/technotes/guides/jdbc/

\(^{23}\)Authorization handling is evaluated through the second sample scenario. This scenario involves to act upon an account that does not pertain to the creator of the reactive tag. This implies to request appropriate authorizations. For instance, Oscar needed to act on Jon’s and Iker’s Delicious accounts. Hence, each student had to request authorizations from the other two mates. And the other way around. Each student had to grant permissions to his mates.
7. RELATED WORK

Our work aligns with those practices using metadata as prescriptive hints. This section introduces a set of dimensions to compare related work. Comparison is based on both the implementation approach and the expressiveness of the ECA rule. Specifically, the Event dimension refers to which facts can raise an event. The Reaction dimension refers to which actions can be triggered. The Usability dimension alludes to the ways users can define rules. Finally, the Event Broker and Reaction Broker dimensions refer to those technical elements that detect triggers and enact actions, respectively. Findings are depicted in Table 2.

Dandelion focuses on wikis [6]. Wiki articles can be annotated with tags. Dandelion introduces three pre-set “task tags”: Add Task, Send Task, and Reminder. These tags are aimed at promoting collaboration by sending emails among wiki participants and managing access control to wiki sections depending on the status of the task tags (i.e. in progress or done). Similarly, TABASCO promotes collaboration among workmates through spreading, sharing and synchronizing their resources in social websites.

Placeless Document focuses on documents [8]. It is similar to TABASCO in the fundamental principles: (1) metadata are a uniform mechanism for resource management, (2) metadata are user-specific rather than a system property and (3) metadata denotes prescriptive behaviour. In Placeless Document, users attach executable code (serialized Java objects) to documents as properties, and they establish which action invocation on documents execute which properties. Quoting Dourish et al. “By allowing properties to carry executable code, responsible for performing relevant tasks that can achieve the needs expressed by these user concerns, we can use this simple property mechanism to make the document management system active and responsive. In turn, using properties to control these features of system behavior allows a level of uniformity and flexibility that is hard to achieve in a world made up of tens or hundreds of separate applications and control panels.” So far, TABASCO only performs one action: item creation. In a Web2.0 ecosystem, this action can handle a wealth of scenarios. Additional actions can be considered, though semantic heterogeneity represents a main stumbling block.

ActiveTag focuses on tagging sites [10]. Therefore, its scope coincides with that of TABASCO. However, they diverge in both the focus and architecture. ActiveTag detects tags through screen scraping. XPath and regular expressions are used to find the tag of interest. On finding, ActiveTag calls web endpoints whose results are mashuped at the rendered page. By contrast, TABASCO detects tagging activity through PuSH hubs which poll websites’ RSS feeds about new posts containing a reaction tag. When this occurs, the post is abstracted into a SIOC item to facilitate ulterior transformation into other resource items. For ActiveTag, efficiency and mashup are the main drivers. For TABASCO, efficiency is not a problem as the result of tagging does not need to be readily available at the target site, and hence, an RSS approach works fine. By contrast, heterogeneity is a main concern since actions can impact different websites, unlike ActiveTag where mashup actions always impact the accessed tagging site.
### Table 2: Metadata-based reactive systems.

<table>
<thead>
<tr>
<th>System</th>
<th>Event</th>
<th>Reaction</th>
<th>Usability</th>
<th>Event Broker</th>
<th>Reaction Broker</th>
</tr>
</thead>
<tbody>
<tr>
<td>TABASCO</td>
<td>On creating a post with a Reaction Tag ...</td>
<td>... create a new post in other site</td>
<td>ECA rules are defined through a Web interface</td>
<td>PuSH hubs inform about a post creation in a web site</td>
<td>SIOC Drivers call site’s APIs to create posts</td>
</tr>
<tr>
<td>Active Tags</td>
<td>On rendering a web page containing an Active Tag ...</td>
<td>... mash-up it with other web pages</td>
<td>Mashups are defined through a Web interface</td>
<td>TagExtractors detect tags through screen scraping</td>
<td>Calls on Web endpoints return data fragments to be mashed up at the tagging site</td>
</tr>
<tr>
<td>Placeless Documents</td>
<td>On adding/deleting documents/properties ...</td>
<td>... execute the code attached as document properties</td>
<td>Active properties are attached to documents</td>
<td>A Dispatcher calls properties for invoked operations</td>
<td>All operations on documents are performed in kernels</td>
</tr>
<tr>
<td>Dandelion</td>
<td>On adding task tags on a wiki page ...</td>
<td>... send an email to the addressee and control the access</td>
<td>n.a.</td>
<td>Event-driven finite state machine to handle user requests</td>
<td>The system sends emails and control the access to wiki sections</td>
</tr>
</tbody>
</table>

### 8. CONCLUSION

We have presented an approach for users to associate behaviour to “reactive tags”. Interoperability, security and usability are identified as the main challenges. TABASCO is introduced as a proof-of-concept about how these requirements can be addressed: (1) an intensive use of ontologies permits to abstract from the specifics of each website; (2) OAuth accounts for security and (3), defining tag’s operational semantics as item transformation permits to find an appropriate balance between expressiveness and usability. Future work includes providing additional drivers for popular social sites, and adding utilities to analyse community behaviour based on their tagging practices using reactive tags.

### 9. REFERENCES


