# Multi-synchronous Collaborative Semantic Wikis

Charbel Rahhal, Hala Skaf-Molli, Pascal Molli, and Stéphane Weiss

INRIA Nancy-Grand Est
Nancy University, France
{skaf,molli,charbel.rahal,weiss}@loria.fr

**Abstract.** Semantic wikis have opened an interesting way to mix Web 2.0 advantages with the Semantic Web approach. However, compared to other collaborative tools, wikis do not support all collaborative editing mode such as offline work or multi-synchronous editing. The lack of multi-synchronous supports in wikis is a problematic, especially, when working with semantic wikis. In these systems, it is often important to change multiple pages simultaneous in order to refactor the semantic wiki structure. In this paper, we present a new model of semantic wiki called Multi-Synchronous Semantic Wiki (MS2W). This model extends semantic wikis with multi-synchronous support that allows to create a P2P network of semantic wikis. Semantic wiki pages can be replicated on several semantic servers. The MS2W ensures CCI consistency on these pages relying on the Logoot algorithm.

## 1 Introduction

Wikis have demonstrated how it is possible to convert a community of strangers into a community of collaborators producing all together valuable contents. Semantic wikis [7,14,15,3] by introducing semantic annotations within wiki pages, have opened an interesting way to mix Web 2.0 advantages with the Semantic Web approach. However, compared to other collaborative tools, wikis do not support all collaborative editing mode such as offline work or multi-synchronous editing [5,10]. It is not possible to work with a wiki as it is possible to work with popular distributed version control systems (DVCS) [1] such as Git and Mercurial. The lack of multi-synchronous support in wikis is a problematic, especially, when working with semantic wikis. In these systems, it is often important to change multiple pages simultaneous in order to refactor the semantic wiki structure. Without multi-synchronous support all incremental changes will be visible to both end users and semantic request engines. Therefore, they can observe inconsistent states of the wiki as in a database system without transactional support. We can say that existing wikis do not support transactional changes. Moreover, the lack of multi-synchronous support prevents users to work insulated [6] and also prevents semantic wikis to support dataflow oriented workflow. One approach to solve this problem is to synchronize a network of semantic wikis using distributed version control systems. Unfortunately, DVCS do not support data types used in semantic wikis and do not ensure the CCI consistency model [16] required for distributed collaborative environment. Another approach is to integrate the DVCS collaboration model into a semantic wikis and manage synchronization of wikis using a CCI preserving algorithm. This can be achieved by deploying a network of semantic wikis. Every node of the network replicates a subset of semantic wiki pages. Any node of the network can be connected to another using a publish/subscribe model. A node that publishes changes just push operations representing these changes in a feed. A node that subscribed to this feed can pull operations at any time and re-execute remote operations locally. In this paper, we present a new model of semantic wikis called Multi-Synchronous Semantic Wiki (MS2W). This model extends semantic wikis with multi-synchronous support that allows to create a P2P network of semantic wikis based on declarative collaborative community. Semantic wiki pages can be replicated on several semantic servers. The MS2W ensures CCI consistency on these pages relying on the Logoot algorithm [18]. In this paper, we define a MS2W ontology and algorithms. This ontology can be used to compute awareness metrics for working in multi-synchronous environment [11]. Finally, we demonstrate that combining these algorithms with the Logoot algorithm allow to preserve the CCI consistency model.

The paper is organized as follows. Section 2 presents background and related works. Section 3 gives an overview of the multi-synchronouns semantic wikis (MS2W) and shows scenarios of new collaboration mode in semantic wikis. Section 4 develops the data model, the algorithms for MS2W and presents the correction of our algorithms. Section 5 presents the implementation of the MS2W. The last section concludes the paper and points to some directions for future works.

## 2 Background and Related Work

This section gives a brief overview of semantic wikis and shows their drawbacks. Then, it presents the definition and the principals of multi-synchronous collaboration. Finally, it presents the CCI consistency model.

### 2.1 Semantic Wikis

Semantic wikis such as Sematic MediaWiki [7], IkeWiki [14], SweetWiki [3] and SWooki [15] are a new generation of collaborative editing tools, they allow users to add semantic annotations in the wiki pages. These semantic annotations can then be used to find pertinent answers to complex queries. Semantic wikis can be viewed as an efficient way to better structure wikis by providing a means to navigate and answer questions or reason, based on annotations. Semantic wikis are an extension of wiki systems that preserve the same principles of wikis such as simplicity in creating and editing wikis pages. Semantic wikis embed semantic annotations in the wiki content by using Semantic Web technologies such as RDF and SPARQL. In semantic wikis, users collaborate not only for writing the wiki pages but also for writing semantic annotations. Traditionally, authoring semantics and creation ontologies has mainly been in the hand of "ontologists" and knowledge management experts. Semantic wikis allow mass collaboration for creating and emerging ontologies. Some existing semantic wikis allows the emergence of the ontology, MediaWiki(SMW) [7] and SweetWiki [3]. Others use the wikis as a collaborative ontologies editors. IkeWiki [14] aims to create an instance of existing ontologies, while OntoWiki [2] aims the creation of ontology schema. In a semantic wiki, users add semantic annotations to wiki page text to represent relations and properties on this page. In SMW users choose their own vocabularies to type links. For instance, a link between the wiki pages "France" and "Paris" may be annotated by a user as "capital".

| Content of wiki page of "France" | Content of semantic wiki page of "France" |
|---|---|
| France is located in [Europe] | France is located in [locatedIn::Europe] |
| The capital of France is [Paris] | The capital of France is [hasCapital:: Paris] |

These annotations express semantic relationships between wikis pages. Semantic annotations are usually written in a formal syntax so they are processed automatically by machines and they are exploited by semantic queries. In spite of their success, semantic wikis do not support a multi-synchronous work mode. Their current model provides only one copy of a semantic wiki page. The state of a semantic wiki page is always the current visible one on the server, intermediate inconsistent states are always visible. Consequently, transactional changes are not be supported neither the isolated work mode nor off-line editing mode. In previous work, we proposed SWooki [15] the first P2P semantic wiki. It combines the advantages of P2P wikis and semantic wikis. It is based on an optimistic replication algorithm that ensures the CCI consistency model. It supports the off-line work mode and transactional changes. Its main limitations are the total replication and the collaboration model. Every peer of the network hosts a replica of wiki pages and a replica of the semantic store. Users cannot choose the pages that they want to replicate neither the period of synchronization. Modifications are propagated on page saving. The collaborative community is implicit. Users cannot choose to whom propagate modifications. Changes propagation is under the control of system and not the users. All connected peers receive and integrate changes. A disconnected peer will receive the modifications of the others at reconnection thanks to the anti-entropy algorithm.

## 2.2 Multi-synchronouns Collaboration

Dourish wrote [5], *"The notion of multiple, parallel streams of activity, is a natural approach to supporting this familiar pattern of collaborative work. Working activities proceed in parallel (multiple streams of activity), during which time the participants are disconnected (divergence occurs); and periodically their individual efforts will be integrated (synchronization) in order to achieve a consistent state and progress the activity of the group"*.

Multi-synchronous applications are different from synchronous and asynchronous ones by managing multiple streams of activities instead of giving the illusion of one stream. In standard collaborative applications, when a modification is made by one user, it is immediately visible by others. However, in multi-synchronous applications, modifications made by one user is not visible by others. It becomes visible only when a user validates his modifications (commits his changes). A visible change does not imply immediate integration by other activities streams. Concurrent modifications will be integrated only when users will decide it. Allowing deferred validation of changes and divergence have several important advantages and impacts [11]: Parallelization of activities, privacy, invisibility of partial changes and experimental changes. Multi-synchronous authoring tools [5,10] allow simultaneous work in isolation and later integration of the contributions. They are characterized by their abilities to support divergence *i.e* parallel steam of activities on replicated data. SAMS [12] allows team members to work in synchronous, asynchronous or multi-synchronous mode while ensuring the coherence of the shared data. It uses an operational transformation algorithm to maintain the consistency of the shared data. Although it is independent of shared objects types, SAMS requires the design of transformation functions that depend on the type of the shared object. Design of transformation functions is a hard task and errors prone. Currently, there are no transformation functions designed for semantic wiki pages. Distributed Version Control Systems[1] such Git manage files and directories. They are not used to track changes in semantic wiki pages. They ensure causal consistency, this implies that concurrent write operations can be seen in a different order on different machines. In this case, if two sites observe 2 write operations in different order then copies on both sites can diverge. DVCS systems are aware of this problem and delegate the problem to external merge algorithms for managing concurrent operations. However, as existing merge algorithms are not intrinsically deterministic, commutative and associative so convergence cannot be ensured in all cases.

### 2.3 CCI Consistency Model

A distributed collaborative system is composed of unbounded set of peers $P$. Objects modified by the system are replicated into a set of replicas R (with $0 < |R| \leq |P|$). Every replica has the same role and hosted on one peer. A peer can host replicas of different objects. Peers can enter and leave the network arbitrary. Every peer has a unique comparable identifier. The modifications applied on a replica are eventually delivered to all other replicas, then, they are applied locally. According to [16], the system is correct if it ensures the CCI consistency model.

– **Causality:** This criterion ensures that all operations ordered by a precedence relation, in the sense of the Lamport's *happened-before* relation [8] will be executed in same order on every site.

– **Convergence:** The system converges if all replicas are identical when the system is idle (eventual consistency).

– **Intention and Intention preservation:** The intention of an operation is the effects observed on the state when the operation was generated. The effects of executing an operation at all sites are the same as the intention of the operation.

Many algorithms that verify the CCI model have been developed by the Operational Transformation community [16] such as SOCT2, GOTO, COT etc. But only few of them (MOT2 [4]) support the join and leave constraint of P2P network. However, MOT2 algorithm suffers from a high communication complexity. WOOT [13] is the first synchronization algorithm that ensures the CCI consistency and respects the P2P constraints. WOOT is used in Wooki [17] a peer-to-peer wiki and an modified version of WOOT that takes in consideration semantic data is implemented in SWooki [15]. Recently, an optimized version of WOOT called Logoot is published in [18], it ensures convergence and preserves the intentions, if the causality is preserved.

## 3 Multi-synchronous Semantic Wiki Approach

Multi-synchronous semantic wikis allow users to build their own cooperation networks. The construction of the collaborative community is declarative, in the sense, every user declares explicitly with whom he would like to cooperate. Every user can have a multi-synchronous semantic wiki server installed on her machine. She can create and edit her own semantic wiki pages as in a normal semantic wiki system. Later, she can decide to share or not these semantic wiki pages and decide with whom to share.

### 3.1 Multi-synchronous Collaboration Model

The replication of data and the communication between servers is made through *channels* (feeds). The channel usage is restricted to few servers with simple security mechanisms that requires no login and complex access control. Capabilities fit perfectly these requirements [9]. The key point is that channels are read-only for consumers and can be hosted on hardware of users. When a semantic wiki page is updated on a multi-synchronous semantic wiki server, it generates a corresponding operation. This operation is processed in four steps: (1) It is executed immediately against page, (2) it is published to the corresponding *channels*, (3) it is pulled by the authorized servers, and (4) it is integrated to their local replica of the page. If needed, the integration process merges this modification with concurrent ones, generated either locally or received from a remote server.

The system is correct if it ensures the CCI (Causality, Convergence and Intention Preservation) consistency model. Multi-synchronous semantic wikis use the Logoot synchronization algorithm [18] to integrate modifications. Logoot ensures the CCI consistency model. More precisely, Logoot ensures convergence and preserves the intentions of operations if the causality of the operations is preserved.

### 3.2 Collaboration Scenarios

This section presents two scenarios of collaboration in multi-synchronouns semantic wikis. We suppose several professors collaborate together through a semantic wiki to prepare lectures, exercises and exams. Later, they want to make lectures available for students and they want to integrate relevant feedbacks from students.

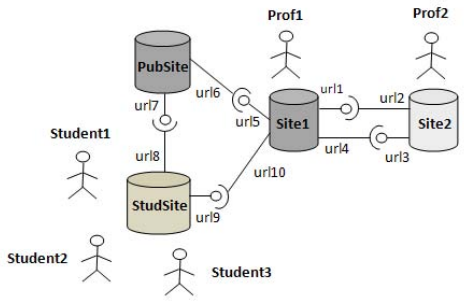


**Fig. 1.** Multi-synchronous semantic wikis scenario

*Scenario 1: Collaboration among professors* For simplicity, we suppose there are two professors $prof_1$ and $prof_2$ (see figure 1). Every professor has her own multi-synchronous semantic wiki, $site_1$ for $prof_1$ and $site_2$ for $prof_2$, respectively.

1. $prof_1$ edits three semantic wiki pages $lesson_1$, $exercises_1$ and $exam_1$ on her site $site_1$. For instance, the page $lesson_1$ edited by $prof_1$ has three lines:

```
  Introduction:
In "multi-synchronous" work, parties work independently in parallel.
[Category::Lesson] [forYear:2009]
```

2. $prof_1$ wants to publish her modifications on $lesson_1$ to $prof_2$. Therefore, she creates a *feed* $url_1$ that contains change set. Finally, she communicates the address of the feed to $prof_2$.
3. $prof_2$ subscribes to this channel and creates a feed $url_2$ to pull modifications. The result is a local semantic wiki page $lesson_1$ that has the same content as $lesson_1$ of $prof_1$.
4. $prof_2$ edits his local copy of the $lesson_1$ page. $lesson_1$ has now five lines:

```
 Introduction:
In "multi-synchronous" work mode, parties work independently in parallel.
[Category::Lesson] [forYear:2009]. This mode is based on divergence and
synchronization phases.
```

5. In her turn, $prof_2$ shares his modifications on $lesson_1$ with $prof_1$. He creates a feed and publishes his modifications. $prof_1$ subscribes to the feed and pulls the modifications published by $prof_2$ and finally integrates these modifications on her local copy of $lesson_1$. The integration process merges the remote modifications with concurrent ones, if any, generated locally by $prof_1$. The integration process has to ensure the convergence of all copies on $lesson_1$ if all generated operations on $site_1$ and $site_2$ are integrated on both sites.

**Table 1.** Multi-synchronous collaboration scenario

| $Site_1$ | $Site_2$ |
|---|---|
| Edit($lesson_1$) | |
| Edit($exercises_1$) | |
| Edit($exam_1$) | |
| CreatePushFeed($f_1$,$q_1$) | |
| | CreatePullFeed($f_1$, $f_2$) |
| Edit( $lesson_1$) | Edit( $lesson_1$) |
| Push($f_1$) | |
| | CreatePushFeed($f_3$,$q_2$) |

The advantages of this collaboration model are, on the one hand, there is no need for centralized server to cooperate. On the other hand, every professor works on her or his own copy in isolation, publishes and pulls changes whenever she wants, *i.e.* the changes propagation is under the control of the user. This collaboration process is not supported in classical semantic wikis. The table 1 represents a part of the scenario.

*Scenario 2: Collaboration among professors and students* In this scenario, $prof_1$ wants to make $lesson_1$ available for her students while she continues to make corrections and minor modifications on $lesson_1$. In order to provide her students with the courses, $prof_1$ publishes them on a public site *pubSite* that can be either her proper public wiki site or the site of the university. *pubSite* is accessible by the students for read only to maintain the courses consistency. However, $prof_1$ manages to integrate relevant feedbacks from students provided by the students sites.

Professor can make continuous improvement of the lessons and can make continuous integration of the modifications and make only consistent modifications visible. Without multi-synchronous support all incremental changes will be visible to both end users and to semantic request engines. Every participant professor and student want to control the visibility of their modifications and want to control the integration of others' modifications.

## 4 Multi-synchronous Semantic Wiki System

This section presents the data model and the algorithms for multi-synchronous Semantic Wiki systems (M2SW). The data model is defined an ontology. Therefore, it is possible to querying and reasoning on the model itself and make future extension. For instance, it is possible to make queries like: (1) "list all unpublished changes", (2) "list all published changes on a given channel", (3) "list unpublished change set of a given semantic wiki page", (4) "list all pulled change sets", etc.

### 4.1 MS2W Ontology

The M2SW ontology is defined as an extension of existing ontologies of semantic wikis [3]. In this section, we present the M2SW ontology and detail only its new vocabulary and their properties.

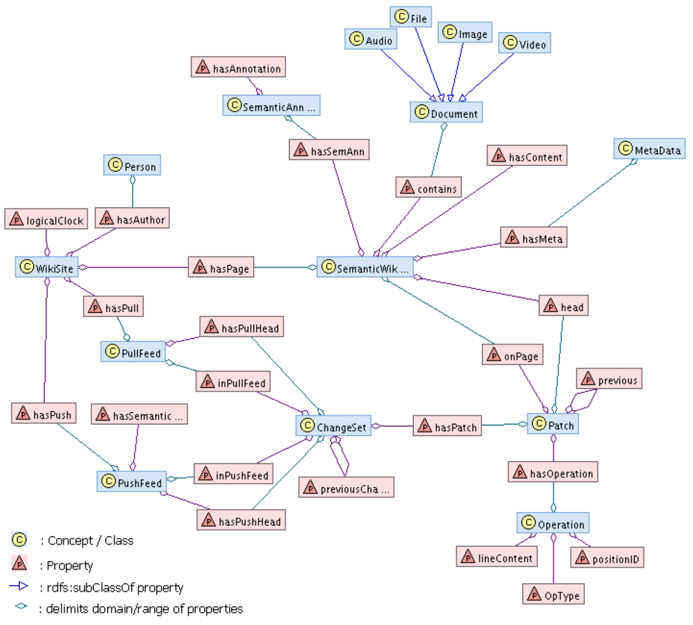


**Fig. 2.** Multi-synchronous ontology

– **WikiSite:** this concept corresponds to a semantic wiki server. A site has the following properties:
  - *siteID* : this attribute contains the *URL* of the site.
  - *logicalClock* : this attribute has a numeric value. Every semantic wiki server maintains a logical clock, this clock is used to identify patches and operations in an unique way in the whole network.
  - *hasPush*, *hasPull* and *hasPage* : the range of these properties are respectively a push feed, a pull feed and a semantic wiki page. A wiki site has several push feeds, pull feeds and several pages .

– **SemanticWikiPage:** this concept corresponds to a normal semantic wiki page. It has the following properties:
  - *pageID* : this attribute contains the *URL* of the page.
  - *hasContent* the range of this property is a String, it contains text and the semantic data embedded in the semantic wiki page.
  - *head* : this property points to the *last patch* applied to the page.

- **Operation:** this concept represents a change in a line of a wiki page. In our model, there are two editing operations : *insert* and *delete*. An update is considered as a delete of old value followed by an insert of a new value. An operation has the following properties:
  - $operationID$: this attribute contains the unique identifier of the operation. $operationID$ is calculated by: $operationID$ = concat($Site.siteID$, $Site.logicalClock++$), the *concat* function concatenates two strings.
  - $opType$: this attribute contains the type of the operation, it can be either an $insert$ or a $delete$.
  - $positionID$: denotes the position of the line in the wiki page. This identifier is calculated by the *Logoot* algorithm[18].
  - $lineContent$: is a string representing text and the semantic data embedded in the line.
- **Patch:** a patch is a set of operations. A patch is calculated during the save of the modified semantic wiki page using the Logoot algorithm. A patch has the following properties:
  - $patchID$: is a unique identifier of the patch. Its value is calculated by : $patchID$ = concat($Site.siteID$, $Site.logicalClock++$)
  - $onPage$: the range of this property is the page where the patch was applied.
  - *hasOperation*: this property points to the operations generated during the save of the page.
  - $previous$: points to the precedent patch.
- **ChangeSet:** a change set contains a set of patches. This concept is important in order to support transactional changes. It allows to regroup patches generated on multiple semantic wiki pages. Therefore, it is possible to push modifications on multiple pages. *ChangeSet* has the following properties:
  - $changSetID$: is a unique identifier of a change set. Its value is calculated as : $changeSetID$ = concat($Site.siteID$,$Site.logicalClock++$)
  - $hasPatch$ property points to the patches generated since the last push.
  - $previousChangeSet$: points to the precedent change set.
  - $inPushFeed$: the range of this property is $PushFeed$. This property indicates the $PushFeed$ that publishes a $ChangeSet$.
  - $inPullFeed$: the range of this property is $PullFeed$ . This property indicates the $PullFeed$ that pulls a $ChangeSet$.
- **PushFeed:** this concept is used to publish changes of a $WikiSite$. It is a special semantic wiki page. It inherits the properties of the *SemanticWikiPage* concept and defines its own properties:
  - $hasPushHead$ : this property points to the *last* published *changeSet*.
  - *hasSemanticQuery*: this property contains a semantic query. This query determines the content of the push feed. For instance, the query can be "find all Lessons", this will return all the pages in the class (category) Lessons. To answer *hasSemanticQuery*, reasoning and querying capabilities of semantic wikis are used.

– **Pull Feed:** this concept is used to pull changes from a remote $WikiSite$. A pull feed is related to one push feed. In the sense that it is impossible to pull unpublished data. A pull feed is also a special semantic wiki page. It inherits the properties of the *Semantic Wiki Page* concepts and defines it own properties:
  - $hasPullHead$: this property points to the last pulled change set pulled.
  - $relatedPushFeed$: this property relates a pull feed to the $URL$ of its associated push feed.

We can extend and build on $MS2W$ ontology. The $MS2W$ ontology is maintained by the $MS2W$ developers. It is defined in OWL DL allowing to querying and reasoning on the patches, ChangeSet, PushFeed, etc. SPARQL can be used to query the MS2W data. For instance, it is possible to list all published patches on a push feed:

$$Published \equiv \exists(hasPatch^{-1}).\exists(inPushFeed^{-1}).PushFeed$$

## 4.2 Algorithms

As any semantic wiki server, a multi-synchronous semantic wiki server defines a *Save* operation which describes what happens when a semantic wiki page is saved. In addition, we define special operations : *CreatePushFeed*, *Push*, *CreatePullFeed*, *Pull* and *Integrate* for the multi-synchronous semantic wiki. We use the Logoot algorithm [18] for the generation and the integration of the *insert* and *delete* operations. In the following, detail these operations for a semantic wiki server called *site*.

**Save Operation.** During the saving a wiki page, the *Logoot* algorithm computes the difference between the saved and the previous version of the page and generates a patch. A *patch* is a set of delete and insert operations on the page ($Op = (opType, operationID, positionID, lineContent)$). *Logoot* calculates the $positionID$, $lineContent$ and the $opType$ of the operation. These operations are integrated locally and then eventually published on a push feed.

```
On Save(page : String, $\overline{page}$:String) :
  Patch(pid=concat(site.siteID, site . logicalClock ++))
  foreach op ∈ Logoot(page, $\overline{page}$) do
    Operation(opid=concat(site.siteID, site . logicalClock++))
    hasOperation(pid,opid)
  endfor;
  previous(pid,page.head)
  head(page,pid)
  onPage(pid,page)
```

**CreatePushFeed Operation.** The communication between multi-synchronouns semantic wiki servers is made through feeds. The *CreatePushFeed* operation creates of a push feed. A push feed is a special semantic wiki page that contains a query that specifies the pushed data. It is used to publish changes of a wiki server. Authorized sites can access the published data. *CreatePushFeed* operation calls the *Push* operation.

```
On CreatePushFeed(name:String,request:String):
  PushFeed(name)
  hasSemanticQuery(name,request)
  hasPush(site,name)
  call Push(name)
```

**Push Operation.** This operation creates a change set corresponding to the pages returned by the semantic query and adds it to the push feed. Firstly, the semantic query is executed, then the patches of the pages returned by the query are extracted. These patches are added to the change set if they have not been published on this push feed yet.

```
On Push(name:String):
    ChangeSet(csid=concat(site.siteID,site.logicalClock++))
    inPushFeed(csid, name)
    let published ← { $\exists$x $\exists$y $\wedge$ inPushFeed(y,name) $\wedge$ hasPatch(y,x) }
    let patches ← { $\exists$x $\forall$p $\in$ execQuery(name.hasSemanticQuery) $\wedge$ onPage(x,p)}
    foreach patch $\in$ {patches $-$ published} do
      hasPatch(csid, patch)
    endfor
    previousChangeSet(csid, name.hasPushHead)
    hasPushHead(name,csid)
```

**CreatePullFeed Operation.** As the replication of data and the communication between multi-synchronouns semantic wiki servers are made through feeds, pull feeds are created to pull changes from push feeds on remote peers to the local peer (cf figure 3). A pull feed is related to a push feed. In the sense that it is impossible to pull unpublished data.

```
On CreatePullFeed(name:String, url:URL)
    PullFeed(name);
    relatedPushFeed(name,url)
    call Pull(name);
```

**Fig. 3.** CreatePullFeed operation

```
On ChangeSet get(cs : ChangeSetId ,url)
    if $\exists$x previousChangeSet(cs,x)
        return x
    else return null;
```

**Fig. 4.** get a ChangeSet operation

**Pull Operation.** This operation fetches for published change sets that have not pulled yet (cf figure5). It adds these change sets to the pull feed and integrate them to the concerned pages on the pulled site.

**get Function.** This function allows to retreive a ChangeSet (cf figure 4).

**Integration Operation.** The integration of a change set is processed as follows (cf figure 6). First all the patches of the change set are extracted. Every operation in the patch is integrated in the corresponding semantic wiki page thanks to the Logoot algorithm.

```
On Pull(name:String):
    while ((cs ← get(name.headPullFeed,
            name.relatedPushFeed) ≠ null)
    let p ← {∃x ∧ inPushFeed(x,name)}
    if cs ∉ {p} then
        inPullFeed(cs,name)
        call Integrate(cs)
    endif
    hasPullHead(name,cs)
  endwhile
```

**Fig. 5.** Pull Operation

```
Integrate(cs:ChangeSet):
  foreach patch ∈ cs do
    previous(patch,patch.onPage.head)
    head(patch.onPage,patch)
    foreach op ∈ hasOperation.patch
        do call logootIntegrate(op)
     endfor
   endfor
```

**Fig. 6.** IntegrateOperation

## 4.3 Correction Model

**Theorem 1.** *Our algorithms ensure the causality.*

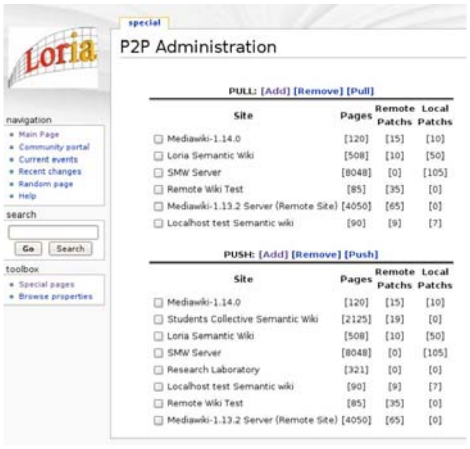
*proof.* A site always respects the causality of its own operations since it generates these operations alone, so these operations placed in the push feed are causal. In the following, we take an hypothesis that a site $S$ pulls from $N$ sites that ensure the causality and we want to prove that the site itself ensures the causality. We will prove that the site $S$ cannot violate the causality through the contradiction proof. Let us consider that the site $S$ violates the causality, i.e. in the content of its push feed there is an operation $op_2$ that depends causally (Lamport dependence) on an operation $op_1$ and $op_2$ is placed before $op_1$ in that feed. The operations in a push feed has two different sources either they are local generated by the site itself or the site had received them from other sites. Let us study both cases (1) *local operations*: the local operations always respect the causality, so a site is not going to violate the causality. (2) *remote operations:* In order to get the operation $op_2$ placed at the position $N$ in the push feed of site $S$, $S$ must pull, i.e. copy all the operations that precedes $op_2$. Getting $op_2$ without having $op_1$ means that $op_1$ was not among the $N-1$ operations that precedes $op_2$ in the push feed which violates the hypothesis that the site $S$ pulls only from $N$ sites that respect the causality. Consequently, it is not possible to find $op_2$ before $op_1$. Therefore, a site always respects the causality alone. If this site pulls from $N$ sites that ensure the causality, it ensures the causality itself. In recurrence, all site ensure the causality for any $N$, where $N$ is an integer. Hence, our algorithm ensures the causality for all the sites in the network.

**Theorem 2.** *Our algorithms ensure the CCI model (Causality, Convergence, Intention).*

*p*roof. Our algorithm is based on Logoot algorithm for the generation and the integration of the operations. It is proved in [18], that the *Logoot* algorithm ensures the convergence and preserves the intentions, if the causality is preserved. Since our algorithms ensure the causality (theorem 1), hence it ensures also the convergence and preserves the intentions.

## 5 Implementation

We are currently implementing the MS2W model as an extension of Semantic MediaWiki [7]. Feeds, ChangeSets and Patches are represented as special semantic wiki pages. They stored in a special namespace to prevent user modification. The Logoot algorithm has been implemented in PHP and integrated in Mediawiki relying on the hook mechanism. The push, pull, createPushFeed and createPullFeed operations are available in special administration pages of MediaWiki as shown in figure 7. This extension is designed to respect the simplicity of the wikis while supporting the MS2W model and the result is an easy way to construct a P2P network of semantic wikis based on Semantic MediaWiki.



P2P Administration

PULL: [Add] [Remove] [Pull]

| Site | Pages | Remote Patchs | Local Patchs |
|---|---|---|---|
| ☐ Mediawiki-1.14.0 | [120] | [15] | [10] |
| ☐ Loria Semantic Wiki | [508] | [10] | [50] |
| ☐ SMW Server | [8048] | [0] | [105] |
| ☐ Remote Wiki Test | [85] | [35] | [0] |
| ☐ Mediawiki-1.13.2 Server (Remote Site) | [4050] | [65] | [0] |
| ☐ Localhost test Semantic wiki | [90] | [9] | [7] |

PUSH: [Add] [Remove] [Push]

| Site | Pages | Remote Patchs | Local Patchs |
|---|---|---|---|
| ☐ Mediawiki-1.14.0 | [120] | [15] | [10] |
| ☐ Students Collective Semantic Wiki | [2125] | [19] | [0] |
| ☐ Loria Semantic Wiki | [508] | [10] | [50] |
| ☐ SMW Server | [8048] | [0] | [105] |
| ☐ Research Laboratory | [321] | [0] | [0] |
| ☐ Localhost test Semantic wiki | [90] | [9] | [7] |
| ☐ Remote Wiki Test | [85] | [35] | [0] |
| ☐ Mediawiki-1.13.2 Server (Remote Site) | [4050] | [65] | [0] |

**Fig. 7.** MS2W Administration page

## 6 Conclusion and Future Work

In this paper, we propose a new model of semantic wikis called Multi-Synchronous Semantic Wiki (MS2W). MS2W extends a semantic wiki with multi-synchronous capabilities. Multi-synchronous semantic wikis allow users to build their own cooperation networks. The construction of the collaborative community is declarative. Every user declares explicitly with whom he would like to cooperate. The replication of the wiki pages on semantic wikis servers and the synchronization periods are variant and under the control of the users. The MS2W model enhances the existing semantic wikis by supporting transactional changes and the off-line work mode. Hence, multiple dataflow oriented workflows can be supported. In addition, the model takes natural advantages of a P2P network, i.e. faults-tolerance, better scalability, infrastructure cost sharing and better performance. In the paper, we develop an ontology for MS2W. Each ontological element in this wiki is exploitable, consequently advanced reasoning and querying capabilities are provided. We develop also a set of algorithms for the creation and the replication of semantic wiki pages and for the propagation and the synchronization of changes between the peers. The MS2W algorithm ensures the CCI consistency on the pages. We are currently implementing the MS2W model as an extension of Semantic MediaWiki. As for future works, we want to support the MS2W with two crucial features: (1) an undo mechanism to facilitate the undo of undesired merge results, (2) an advanced mechanism for divergence metrics based on querying the semantic annotations.

## References

1. Allen, L., Fernandez, G., Kane, K., Leblang, D., Minard, D., Posner, J.: ClearCase MultiSite: Supporting Geographically-Distributed Software Development. In: Software Configuration Management: Scm-4 and Scm-5 Workshops: Selected Papers (1995)
2. Auer, S., Dietzold, S., Riechert, T.: OntoWiki – A tool for social, semantic collaboration. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 736–749. Springer, Heidelberg (2006)
3. Buffa, M., Gandon, F.L., Ereteo, G., Sander, P., Faron, C.: Sweetwiki: A semantic wiki. Journal of Web Semantics 6(1), 84–97 (2008)
4. Cart, M., Ferrie, J.: Asynchronous reconciliation based on operational transformation for p2p collaborative environments. In: International Conference on Collaborative Computing: Networking, Applications and Worksharing (2007)
5. Dourish, P.: The parting of the ways: Divergence, data management and collaborative work. In: 4th European Conference on Computer Supported Cooperative Work (1995)
6. Estublier, J., et al.: Impact of the research community on the field of software configuration management: summary of an impact project report. SIGSOFT Softw. Eng. Notes 27(5), 31–39 (2002)
7. Krötzsch, M., Vrandecic, D., Völkel, M., Haller, H., Studer, R.: Semantic wikipedia. Journal of Web Semantic 5(4), 251–261 (2007)
8. Lamport, L.: Times, Clocks, and the Ordering of Events in a Distributed System. Communications of the ACM 21(7), 558–565 (1978)
9. Levy, H.: Capability-Based Computer Systems. Butterworth-Heinemann Newton, MA (1984)
10. Molli, P., Skaf-Molli, H., Bouthier, C.: State Treemap: an Awareness Widget for Multi-Synchronous Groupware. In: Seventh International Workshop on Groupware - CRIWG. IEEE Computer Society, Los Alamitos

11. Molli, P., Skaf-Molli, H., Oster, G.: Divergence awareness for virtual team through the web. In: Integrated Design and Process Technology, IDPT (2002)
12. Molli, P., Skaf-Molli, H., Oster, G., Jourdain, S.: Sams: Synchronous, asynchronous, multi-synchronous environments. In: Seventh International Conference on CSCW in Design (2002)
13. Oster, G., Urso, P., Molli, P., Imine, A.: Data Consistency for P2P Collaborative Editing. In: Conference on Computer-Supported Cooperative Work (2006)
14. Schaffert, S.: IkeWiki: A Semantic Wiki for Collaborative Knowledge Management. In: 1st Workshop on Semantic Technologies in Collaborative Applications (2006)
15. Skaf-Molli, H., Rahhal, C., Molli, P.: Peer-to-peer semantic wikis. In: DEXA 2009: 20th International Conference on Database and Expert Systems Applications (2009)
16. Sun, C., Jia, X., Zhang, Y., Yang, Y., Chen, D.: Achieving Convergence, Causality Preservation, and Intention Preservation in Real-Time Cooperative Editing Systems. ACM Transactions on Computer-Human Interaction 5(1) (1998)
17. Weiss, S., Urso, P., Molli, P.: Wooki: A P2P wiki-based collaborative writing tool. In: Benatallah, B., Casati, F., Georgakopoulos, D., Bartolini, C., Sadiq, W., Godart, C. (eds.) WISE 2007. LNCS, vol. 4831, pp. 503–512. Springer, Heidelberg (2007)
18. Weiss, S., Urso, P., Molli, P.: Logoot: a scalable optimistic replication algorithm for collaborative editing on p2p networks. In: ICDCS. IEEE, Los Alamitos (2009)