

# Connecting Topincs

## Using transclusion to connect proxy spaces

Robert Cerny

Anton-Kubernat-Strasse 15, A-2512 Oeynhausen, Austria  
robert@cerny-online.com  
<http://www.cerny-online.com>

**Abstract.** Topincs is a software system for agile and distributed knowledge management on top of the common web infrastructure and the Topic Maps Data Model. It segments knowledge into stores and offers users a way to establish a temporary connection between stores through transclusion and merging. This allows to easily copy topics and statements. Through this mechanism, later acts of integration are simplified, due to matching item identifiers. Furthermore, transclusion can be used to create permanent connections between stores.

## 1 Introduction

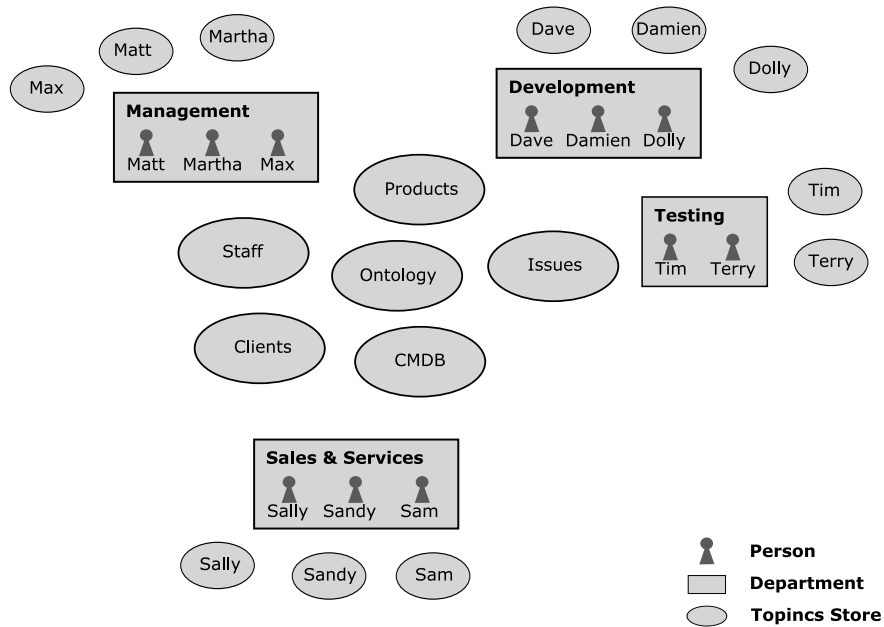
A Topincs store [5, 6, 10] is a knowledge repository for an individual or a group to make statements about subjects. It uses the Topic Maps Data Model [7] for representing knowledge. Throughout this paper we will illustrate crucial aspects by the example of a small fictional software company, called Modestsoft. It employs around 10 people and has various departments, including development, testing, and sales & services.

Modestsoft uses several shared Topincs stores to manage its organizational knowledge. Those stores deal with Modestsoft's *staff*, *products*, *issues*, *clients*, and their *configurations*, to name only a few. Additionally, every employee has his own store. While the shared stores have a clear boundary around their domain, the personal stores may contain topics and statements of a wider variety and are considered a repository for note taking, and as such they function as a memory extender [4]. Figure 1 illustrates Modestsoft's setup and the Topincs stores they use to manage their spontaneous knowledge recording demands.

A Topincs store qualifies as a knowledge node in a distributed knowledge management system as laid forth by Bonifacio et al. in [3]. It offers *semantic autonomy* for a person or group to be able to encode knowledge in a topic map. Topincs is a server application exposing a web interface. The current implementation uses PHP to process requests, runs on top of Apache, and uses MySQL for persistence. The two existing clients are browser-based JavaScript applications that address different user needs, but operate on the same repository:

- Topincs Editor: A topic map editor which offers maximum expressivity and requires in-depth knowledge of the Topic Maps paradigm. It is necessary to state prototypical statements that pave the way for ontological reflection.

- Topincs Wiki: A semantic wiki for quick editing with limited expressivity for people with intermediate computer skills and little to no knowledge of Topic Maps.



**Fig. 1.** A schematic overview of the knowledge landscape of Modestsoft, a fictional software company with agile and distributed knowledge management.

Both clients offer specialized input elements to ease the creation of statements and make the topic map editing process similar to filling out a form. Yet, the Editor has a strong affiliation to the Topic Maps terminology which makes it unusable for most people. The Wiki avoids such terminology and allows only statements of a known form and is therefore usable by a wider audience. It extracts a topic map fragment for the subject of the Wiki page from the store by an algorithm similar to the one defined by Ahmed [1] and renders this fragment in the browser for viewing and editing. New statements about a subject can be issued one by one. The next step in this evolution of restricting expressivity leads to Topincs Forms, which allows the grouping of statements and simple validation for more control over what should and can be said.

Topincs currently uses *ontological reflection* to suggest statement types to the user and to determine the datatype of an occurrence [6]. In a later version it will be instrumented to restrict the players of roles to topics of certain types.

This mechanism makes Topincs a tool for agile knowledge management where encoding demands can be satisfied within minutes as they occur, without the need for detailed planning and programming. A user simply has to open the Editor, which has no restrictions and allows him to issue the new prototypical statement. Subsequently, due to *ontological reflection*, the statement type will be suggested in the Wiki. One Topincs installation can host any number of stores which are identified by an URL relative to the domain. Modestsoft uses `/staff`, `/products`, `/clients`, and `/cldb` for its shared stores.

This paper discusses a solution to easily exchange topics and statements between stores with the help of transclusion. This exchange allows *coordination* between knowledge nodes, which, besides *semantic autonomy*, is demanded as the second criterion for a distributed knowledge management system by Bonifacio et al. [3]. Without this solution Modestsoft had to create the topic type *Person* manually in every shared and group store and assign the subject identifier every time. With the work presented in this paper it is possible to create all typing topics in one designated ontology store and copy it into other stores. Once a topic is copied, it is easy to update it and fetch new statements.

The isolation of the proxy space is necessary to enable a search with acceptable response time when users need to refer to a subject they have in mind. It has the additional benefit that the user is forced to create a proxy for the subject which is under his control. By creating an item, Topincs automatically assigns an item identifier, an URL, which can be used to retrieve and manipulate the item by the HTTP methods POST, GET, PUT, and DELETE [5].

By using a Published Subject Identifier (PSI), the fact that two topics represent the same subject can be established manually. This paper does in no way intend to discourage the use of PSIs, on the contrary, it presents an alternative that relieves the users of the manual assignment and still lets them benefit from the mechanism that the Topic Maps Data Model provides for establishing sameness of subject for two proxies, by the means of item identifiers [7].

This paper introduces three forms of connecting Topincs stores and illustrates them with use cases. Furthermore it discusses the problems that arise with this approach and gives an outlook on future work.

## 2 Temporary Transclusion

Ted Nelson coined the term *transclusion* in his book *Literary Machines* [9]. Transclusion means the composition or augmentation of a document by including other documents or parts thereof by reference. It can be applied to documents that are written in natural language or a markup language. In both cases, it is important that the transcluded fragments can be interpreted, even though they are taken out of their original context. In our Topic Maps use case, it is best described by a transient merge map statement which triggers semantic integration of topic maps with some small differentiations depending on the type of transclusion.

The application of temporary transclusion in Topincs is the following: A user browses a foreign Topincs Wiki and discovers a page about a subject, for which

a topic in his Topincs store exists. He decides to temporarily include the foreign page into his own page in order to see both merged. The result of this process is his page about the subject augmented with the foreign statements, which can be recognized visually. This temporary transclusion has no effect on the proxy space, yet it forms the base for two stronger connection types, namely *permanent transclusion* and *item copying*.

### 3 Permanent Transclusion

This form of connection allows the user of a store to permanently connect one topic to another topic representing the same subject in a different store. Whenever someone browses the page with the permanent transclusion directive, the included topic map will be retrieved and merged in the web browser at access time. The user will be able to recognize the foreign statements. Clicking on the player of a foreign association will lead the user into the respective proxy space given that there is no proxy in the local store. Updates in the foreign page will be reflected after a page reload.

The number of permanent transclusion directives on a page is not restricted. It is also possible to transclude arbitrary structured information in a page by incorporating a transformation step in the process.

### 4 Item Copying

From any form of transclusion, a user with edit rights on the Topincs Store can decide to save foreign statements into his proxy space. Since this process is very similar to manually creating statements, the existing infrastructure for editing and saving statements in the Wiki edit panel is used. Yet, the user has also the possibility to merge foreign topics with local ones. With a statement, all topics that are referred to within the statement, e.g. in the properties *type* or *scope*, are copied as well.

By copying statements and topics, the user gains control over the items. He can then form his own statements about the new subjects and even correct copied statements. In the copying process the item identifiers are transferred, so that on subsequent views of the page it is possible to update statements. On a page with permanent transclusion in place, this will happen automatically due to matching item identifiers. Without permanent transclusion, the user has to manually initiate the update.

In particular, the copying of topics of the ontology is beneficial for later integration of information between the two stores, since the need for manual merging will be significantly less and any act of transclusion will present a more homogeneous page. Within a Topincs installation, or even over several installations, it is recommended to have a designated store for the ontology, where users can fetch their topic types, occurrence types, name types, association types, and role types. With this store in place, users do not have to worry about later integration, given that the usage of the types is correct. The only practical method to

enforce the correct usage of a proxy must be modeled after the mechanism used for learning of natural language. If the community observes the incorrect usage of a type, it must be corrected, like we correct little children when they are using a word incorrectly.

At a later stage, with Topincs Forms, a stronger enforcement of the correct usage can be in place. Topincs Forms is a framework for developing forms that generate sets of statements that are fed to a Topincs store via the web interface. This addition to Topincs is only in a very early planning phase. Its purpose, though, is clear. While the Topincs Editor offers complete freedom regarding which statements a user can issue, and the Wiki restricts users to statements of a certain form, Topincs Forms will put even stronger restrictions in place on which statements have to be said, and how these statements should relate to each other. This convergence brings us closer to achieving highly structured data which are not only useful for human consumption, but can be consumed by programs as well.

The screenshot shows a web interface for editing a person's profile. At the top, there are buttons for 'view', 'edit', and 'include'. Below that, the name 'Dolly' is displayed, followed by 'Person'. A toolbar contains '+ New statement', 'Selection', 'Preview', and 'Save'. The main form has several sections:

- Date of birth:** 06/23/1978 (with a calendar icon), marked as 'New' (N) and 'Saved' (checkmark).
- Topic name:** Dolly (with a document icon), marked as 'Deleted' (trash icon).
- Instance of:** Person (with a document icon), marked as 'Deleted' (trash icon).
- Reports to:** Dave (with a document icon), marked as 'New' (N) and 'Saved' (checkmark).
- Has skill:** A list of skills: Java, Ruby, Unix, and Windows. Each skill entry has a 'New' (N) icon and a 'Saved' (checkmark) icon.
- Employed by:** Modestsoft, Inc. (with a document icon), marked as 'New' (N) and 'Saved' (checkmark).
- Has assignment:** Refactor all literal strings into constants (with a document icon), marked as 'New' (N) and 'Deleted' (trash icon).

**Fig. 2.** Dave transfers statements about Dolly from the staff store into the issues store. New statements are marked by a white N (for *New*) on blue background. All foreign statements from the staff repository have a grey background. All statements local to the issues store have a white background. Dave has selected uninteresting statements for deletion and already assigned Dolly her first issue.

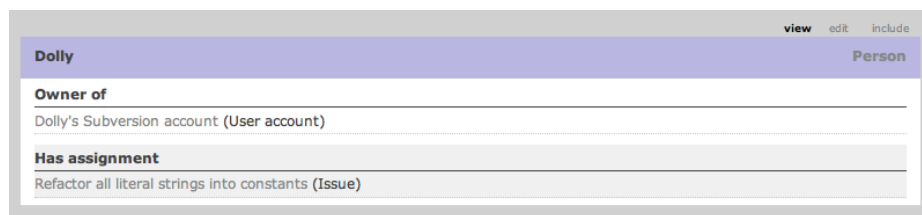
## 5 Use cases

### 5.1 Temporary transclusion and item copying

Our lead developer, Dave, wants to assign an issue to his newest team member Dolly. On the person page of the staff store, he can quickly transfer the statement about Dolly's existence into the issues store where assignments are recorded. On the include panel of Dolly's page in the issues store he can request the transclusion of the topic map about Dolly in the staff store. Dave switches to the edit panel and deletes those statements about Dolly that are not interesting in the context of software issues, e.g. Dolly's date of birth, and keeps the rest, e.g. which programming languages Dolly knows. He manually creates the issue assignment statement. Figure 2 shows what Dave has done so far. He finishes the transfer by saving the page. From now on there is a topic representing Dolly in the issues store and this proxy holds references to other proxies that represent the same subject. The transclusion of Dolly's staff store topic map can be requested by Dave on the include panel of her Wiki article at any time to see whether statements were corrected or new statements were added.

### 5.2 Permanent transclusion and agile extension

Dolly is an experienced programmer. Developers in her old company used the issue tracking system to record the time they spent on issues. She knows the planning benefits of such information and wants to do the same thing in Topincs. She is not quite sure yet how to do it, so she decides to do a test run in her personal store. She visits her article page in the issues store and temporarily includes her page into the one in her personal store. She makes this transclusion permanent, so that she can easily copy new issues into her store in order to make effort tracking statements about issues assigned to her. This permanent transclusion allows her to learn about new assignments in her own personal store, although they are made by Dave in the issues store. Figure 3 shows her page in her personal store.



**Fig. 3.** Dolly's article in her own store. She has already noted that she owns a Subversion account. The assignments are retrieved from the issues store by permanent transclusion. Even the association type *Has assignment* is indicated as foreign with a grey background.

She analyses which topic and statement types are necessary for a minimal effort tracking system and comes up with the following: one topic type *Work session*, three occurrence types *Start time*, *End time* and *Summary*, and two association types *Participation* and *Realization*. The first association type is to relate a work session to the worker and the latter is to relate a work session to the issue that is processed. The creation of these types takes her approximately fifteen minutes including the issuing of the prototypical, pre-reflection statements in the Editor. She makes it a habit to start work by creating a work session with the statements *Start time*, *Participation*, and *Realization* and to finish it by stating the *Summary* and *End time*. Later she tells her boss Dave about her extension and he decides that all developers should track their efforts. In order to keep concerns separated, he creates a new Topincs store `/efforts` and asks Dolly to copy her Work sessions into the Store. Due to ontological reflection, other users have immediate access to the statement types for work sessions in the Wiki.

## 6 Problems

By using transclusion we have gained an easy, quite simple way to copy items from one store to another. This handy mechanism, however, enlarges one problem which already exists within one repository: how to enforce the correct interpretation of a topic. Our lead developer Dave creates an occurrence type *Resolving date*. He intends it to be used by his developers to state that an issue was resolved on a certain date. He has to communicate his intent by creating a description. Within Modestsoft, the existence of such a type and its meaning can be announced in meetings. All humans, whether they consume the occurrence type on the Wiki page of an issue or whether they write programs using this type, participate in a social contract on when and how to use this topic. If the proxy leaves the boundaries of this social contract, in our case the office of Modestsoft, the likelihood of misinterpretation increases.

While this is a serious problem, it is one that affects any form of user interface. When completing a form, a user encounters as many questions as the form has fields. He interprets a question and, based on this interpretation, completes the field. Organizations use trainings to ensure the correct interpretation of the questions that their software asks users. It is hard to imagine a process with less control that achieves the same result. The problem magnifies if one considers global knowledge interchange. The domain has no boundaries — *people can talk about anything* — and users do not come from one cultural background and thus do not share an implicit context which supports them in their interpretation.

Besides unintentional misinterpretation of topics, there is the possibility of intentional misinterpretation. Something similar can frequently be observed in long running projects using relational databases: The meaning of a table column is altered or enriched. Usually, this act is justified by the high organizational costs of introducing new columns. Since the costs of introducing new statement

types in a Topic Maps system are much lower and distinct proxies can always be merged, such intentional misinterpretation should be avoided at any price.

With the presented mechanism, it is very easy for a Topincs user to declare subject sameness for two proxies. Unfortunately, a wrong decision is difficult, nearly impossible to revert in an early merging approach that Topincs uses since incoming statements always use the current subject identity status to decide which topic a statement is attached to. Even if erroneous subject sameness is detected and corrected, the statements about the *other subject* remain at the incorrect topic. A system with late merging, where all topics are considered distinct when storing a statement, would not have this issue.<sup>1</sup>

The current transclusion mechanism is implemented by requesting a topic map in Json Topic Maps format [5] using the JavaScript object `XMLHttpRequest` in the Wiki article page. Due to security restrictions of the web browser this only succeeds within the same domain. Since this issue affects many Ajax applications, there are several work-arounds available and a standard approach for Cross-Domain Ajax requests is work in progress.<sup>2</sup>

## 7 Conclusion and Future Work

Due to the methods presented in this paper, Topincs stores are no longer isolated islands of knowledge, but offer mechanisms for the exchange of topics and statements. Topincs users can benefit from the integration mechanism of the Topic Maps technology without using PSIs, but rather by a simple item copy procedure. By copying topic and statement types the manual work necessary to create a distributed knowledge landscape and to integrate information is significantly reduced.

The declaration of subject sameness of a foreign topic and local topic in the store where the transclusion takes place should be supported by an automated detection mechanism. Maicher's SIM Approach [8] looks very promising in this respect.

Since established organizations have most of their information stored in arbitrary formats we would like to adopt Barta's idea of resource wrappers producing virtual maps [2]. In our case, such virtual maps would ideally contain item identifiers, so that statements can be recognized. A partial satisfaction of the Topincs Web Service Interface [5] regarding reading requests on item identifiers would allow even tighter integration of such virtual maps into Topincs.

With the exchange mechanism in place, the manual labor to manage personal or organizational knowledge in multiple Topincs stores is reduced since topics and statements can simply be copied from one repository to another, maintaining their identity, and easing later integration tasks. Nonetheless, Topincs should be made more user friendly, by further lowering the requirements to encode knowledge in a Topincs store. The most promising approach to this issue is the

---

<sup>1</sup> Personal conversation with Lutz Maicher and Benjamin Bock in Leipzig in February 2008.

<sup>2</sup> <http://www.w3.org/TR/access-control/>

introduction of forms. Modestsoft chief tester Tim and his team always have to issue the same group of statements when they create an issue: *Reported by*, *Reporting date*, *Affects component*, *Discovered in version*, and *Description*. When a developer fixes a bug he has to state: *Resolved by*, *Resolving date*, *Resolved in version*, and *Comment*. In the Wiki, these statements have to be issued one by one, which is cumbersome in such a frequent task. Two new features will ease this activity: 1) statements can be grouped to forms and 2) ontological reflection on role types will allow the presentation of players in a select box, a control that is more widely known than the current completion widget. Both planned features will help Topincs users to create more homogeneous topic maps that allow programmatic processing with less effort and without specifying a schema.

## References

1. Ahmed, K.: TMSHare – Topic Map Fragment Exchange In a Peer-To-Peer Application. In: Proceedings of XML Europe 2002, (2003).
2. Barta, R.: Knowledge-Oriented Middleware Using Topic Maps. In: Maicher, L.; Garshol, L.M.: Proceedings of the Third International Conference on Topic Maps Research and Applications (TMRA'07), Leipzig; Springer LNAI 4999, (2008).
3. Bonifacio, M.; Bouquet, P.; Cuel, R.: Knowledge Nodes: the Building Blocks of a Distributed Approach to Knowledge Management. In: Journal of Universal Computer Science 8(6):191-200, (2002).
4. Bush, V.: As we may think. The Atlantic Monthly, July 1945. Reprinted in: Interactions (III) 2: 35-46, (1996).
5. Cerny, R.: Topincs – A RESTful Web Service Interface For Topic Maps. In: Maicher, L.; Sigel, A.; Garshol, L.M.: Proceedings of the Second International Conference on Topic Maps Research and Applications (TMRA'06), Leipzig; Springer LNAI 4438, (2007).
6. Cerny, R.: Topincs Wiki – A Topic Maps Powered Wiki. In: Maicher, L.; Garshol, L.M.: Proceedings of the Third International Conference on Topic Maps Research and Applications (TMRA'07), Leipzig; Springer LNAI 4999, (2008).
7. ISO/IEC IS, 0-2:2006: Topic Maps – Data Model, International Organization for Standardization, Geneva, Switzerland (2006). <http://www.isotopicmaps.org/sam/sam-model/>
8. Maicher, L.: Topic Map Exchange in the Absence of Shared Vocabularies. In: Maicher, L.; Park, J.: Proceedings of the First International Workshop on Topic Maps Research and Applications (TMRA'05), Leipzig; Springer LNAI 3873, (2006).
9. Nelson, T.H.: Literary Machines. Sausalito; Mindful Press, (1981).
10. Sigel, A.: Report on the Open Space Sessions. In: Maicher, L.; Park J.: Proceedings of the First International Workshop on Topic Maps Research and Applications (TMRA'05), Leipzig; Springer LNAI 3873, (2006).