

# Many Views, Many Modes, Many Tools... One Structure: How Do XooML-based Tools Work?

William Jones  
The Information School  
University of Washington  
Mary Gates Hall, Suite 370  
Seattle, WA 98195-2840 USA  
williamj@uw.edu

Kenneth M. Anderson  
Department of Computer Science  
University of Colorado  
430 UCB  
Boulder, CO 80309-0430 USA  
kena@cs.colorado.edu

## ABSTRACT

This demonstration complements the accepted paper, “Many Views, Many Modes, Many Tools... One Structure: Towards a Non-disruptive Integration of Personal Information” by delving deeper into backend use of the XML-based *XooML* schema as illustrated through its current use in three separate tools: Planz, QuickCapture and FreeMindX.

## Categories and Subject Descriptors

H.5.M. [Information Systems]: Information Interfaces and Presentation: Miscellaneous.

## General Terms

Design

## Keywords

Personal information management, PIM, XML schemas, application integration, open hypermedia, structural computing.

## 1. INTRODUCTION

XooML<sup>1</sup> (for Cross (X) Tool Mark-up Language) takes a step towards addressing a basic tension in the development of supporting tools of Personal Information Management (PIM) and, more generally, in the development of computer-based tools for end users: How to innovate without forcing people to re-organize or re-locate their information? XooML takes aim on a vision of information management: One integrative structure for the organization of information; many tools in support of this structure, its creation and its life-long elaborative use.

The schema in its design and use builds on the lessons learned by the open hypermedia and structural computing communities while moving forward with new techniques that address some of the changes introduced by the evolution of the term “application” to move beyond desktop apps to mobile apps, cloud-based apps and various hybrid architectures. The schema also reflects lessons learned in the development of a Planz prototype.

Planz provides document-like overlays to a personal file system in support of a project-based organization of documents and other forms of information including email messages, web references and informal notes.<sup>2</sup> Planz provides the affordances of a basic word processor and outliner. Document-like project plans are

simply an alternate way to view and work with a folder hierarchy in the file system. The headings and subheadings of a plan correspond to folders and subfolders in the file system

Choice points in Planz were analyzed to identify several directions of generalization to be realized in a schema. The biggest direction of generalization extends from one or some tools to many tools. How to develop a schema with provisions for not just one or some tools but many tools? Three requirements follow:

1. The schema needs to provide “growing room” for any number of tools to persist tool-specific settings as attributes.
2. There must be no risk of confusion or collision between the attributes used by different tools.
3. While support for tool-specific settings is essential, there must also be a common set of attributes that all tools can support via an incremental adoption approach. This requirement ensures that multiple tools can visualize the same set of structures in different ways while ensuring that an update performed by any particular tool is compatible with all of the tools making use of the shared attributes.

The first two constraints are met through bundling of tool-specific attributes into special tool-specific sub-elements. A given sub-element and each of its attributes can then be uniquely specified by the tool-specific URI of its namespace declaration.<sup>3</sup> The use of elements to bundle attributes happens at two levels:

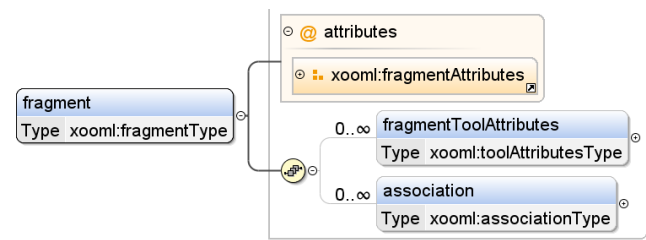


Figure 1. A fragment is a **bundling of “common” (tool-independent) attributes** followed by zero or more bundles of tool-specific attributes (*fragmentToolAttributes*) followed by zero or more *associations*.

Fragment (node) level. For any given information item, the schema defines the structure of an associated fragment of metadata. As depicted in Figure 1, a fragment is a bundling of fragment-level “common” (tool-independent) attributes (which support the third requirement above) followed by zero or more bundles of tool-specific attributes (*fragmentToolAttributes*) followed by zero or more elements of type *association*.

<sup>1</sup> See [kff.ischool.washington.edu/XMLschema/0.41/XooML.xsd](http://kff.ischool.washington.edu/XMLschema/0.41/XooML.xsd).

<sup>2</sup> Planz works under Microsoft Windows and integrates with Microsoft Outlook, Microsoft Word, and other Microsoft Office applications. However, the Planz approach readily extends to other operating systems and other applications.

<sup>3</sup> E.g., `xmlns=http://kff.ischool.washington.edu/xmlns/planz`.

Association (link) level. This pattern of bundling is partially repeated for each association element within a fragment. As depicted in Figure 2, an association is a bundling of common (tool-independent attributes) followed by zero or more bundles of tool-specific attributes.

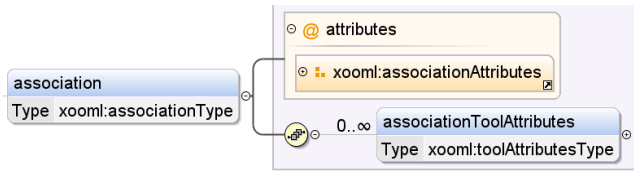


Figure 2. An association is a bundling of common (tool-independent attributes) followed by zero or more bundles of tool-specific attributes.

The demonstration will show the current use of the schema in three separate tools:

- *Planz* (version 8.2) as discussed above.
- *QuickCapture*<sup>4</sup> – a program used to capture a link to an item (document, email message, web page) that appears in the user’s active window. The link appears by default as a shortcut in a “Notes” folder and as an association under the corresponding heading in *Planz*.
- *FreeMindX* – a tool to create mind maps created by “wrapping” the open-source *FreeMind*<sup>5</sup> with schema support

*FreeMindX*, directed to the same “House re-model” project depicted in Figure 1, yields the mind-map view shown in Figure 3. Schema fragments have shared use by each tool and can be easily inspected to see how their shared use works in practice<sup>6</sup>.



Figure 3. The house re-model rendered as a mind map in *FreeMindX*.

It is straightforward to integrate support for this schema into additional tools, as illustrated in Table 1. Tools will all work with bundles of “common attributes” at both the level of a fragment and each of its associations. Each tool can then name and persist values for its own tool-specific attributes at both the fragment level and for each of a fragment’s associations. *Planz*, for example, in order to support an “in-place” archival feature, maintains an *isVisible* attribute for each association. A mind-mapping tool might maintain attributes of “radius” and “polarAngle” per association in order to persist the angle of an outgoing link and the offset of the node to which it points.

Table 1. The attribute bundles of fragment can be grouped by level (fragment-level or association-level) and scope (common or tool-specific).

	Fragment	Association
Common	<i>schemaVersion</i> <i>relatedItem</i> <i>defaultApplication</i> <i>levelOfSynchronization</i> ...	<i>ID</i> <i>associatedItem</i> <i>associatedIcon</i> <i>associatedFragment</i> <i>openWithDefault, ...</i>
Tool-specific (Planz)	<i>toolVersion</i> <i>toolName</i> <i>showAssocsMarkedDone</i> <i>showAssocsMarkedDefer,</i> ...	<i>isVisible</i> <i>isHeading</i> <i>isCollapsed, ...</i>
Tool-specific (FreeMind)	<i>toolVersion</i> <i>toolName, ...</i>	<i>radius</i> <i>polarAngle, ...</i>
Another tool...	...	...

## 2. RELATED WORK

With respect to work performed by the hypermedia community, the approach taken by *Planz* is most compatible with the ideas of structural computing [2] and the lightweight approach advocated by such frameworks as *SmallSC* [1]. The role of structure server is taken over by the combination of the metadata stored in *XooML* fragments and the synchronization service in *Planz*. This service monitors changes to *XooML* fragments (made by the integrated applications) and changes to the file system (made by potentially any application or operating system service) with the sole task of synchronizing the changes between the two. The primary structural concept is that of the multidigraph that is modeled in XML via XML’s own hierarchical document structure augmented by one-way links between nodes of the document or to many other locations via the use of URIs.

## 3. REFERENCES

1. Anderson, K.M. Towards lightweight structural computing techniques with the *SmallSC* framework. *Proceedings of the 2005 symposia on Metainformatics*, ACM (2005), 1.
2. Nürnberg, P.J., Leggett, J.J., and Schneider, E.R. As we should have thought. *Proceedings of the eighth ACM conference on Hypertext*, ACM (1997), 96-101.

<sup>4</sup> *QuickCapture* comes with the installation of *Planz* and can also be installed separately.

<sup>5</sup> [http://freemind.sourceforge.net/wiki/index.php/Main\\_Page](http://freemind.sourceforge.net/wiki/index.php/Main_Page)

<sup>6</sup> Currently, each fragment is stored in a file named “xooml.xml”.