XooML: XML in Support of Many Tools Working on a Single Organization of Personal Information

William Jones
University of Washington
Seattle, WA 98195 USA
williamj@uw.edu

ABSTRACT
XooML 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? Seven considerations in the design of a XooML schema follow from experiences in the iterative evaluation and development of a Planz prototype. Considerations take aim on a vision of PIM: One integrative structure for the organization of personal information; many tools in support of this structure, its creation and its life-long elaborative use.

Categories and Subject Descriptors
H5.m. Information systems: Information interfaces and presentation (e.g., HCI): Miscellaneous.

General Terms
Documentation, Design, Human Factors

Keywords
Personal information management, PIM, XML, RDF.

1. INTRODUCTION
People consistently express a desire for – a yearning for – a greater integration of their personal information [5,6,32]. In a recent study [8], when asked to describe their "ideal system of information management", seventeen of twenty participants described a current state of information fragmentation and explicitly expressed a desire for greater integration with comments such as:

"ideally I would have like a unified system, I wouldn't have all these different databases and all these different check lists and manuals"…"something that unified all of the separate tools and databases that I use" –JS-182, p4.

"..maybe go from media to media a little bit better, …if I store something out on the wiki, it will also store something on, in the, in the file structure." –AP-123, p4.

A unified system. Everything in one place. Integration. But how?

Various proposals and prototypes have been made over the years. Vannevar Bush's Memex would link and organize information through "associative" trails [9]. *Lifestreams* places documents and other information items in a continuous, time-ordered stream [16]. *Presto* offers a uniform property-based platform for information management [14]. *Haystack* [27,28] attempts a uniform encoding of information in RDF (Resource Description Framework) subject-predicate-object "statements"[1] and also introduces the important notion that "anything of interest" might be addressable by a Uniform Resource Identifier (URI)[2]. *MyLifeBits* is able to store a wide variety of information types in a SQL Server database [18].

The integrative potential of personal ontologies is explored through prototypes such as *OntoPIM* [10], *OBIWAN* [11], and *PIA designer* [12] although it is also noted that "… ontological structure is not very clear, especially to a non-expert user with no familiarity with ontologies" [30:6].

Some research efforts approach the goal of information integration by creating a Web-based, shared sub-stratum (e.g., of basic utilities and structured storage) to be used by participating tools (applications, services) in ways analogous to the use by desktop applications of the file system and operating system functions (e.g. [17]). This is approach is much in evidence in efforts to realize an open hypermedia system in which anchors, links and other structural elements are flexibly defined and have existence independently of the documents to which they apply (e.g., [2,4,13,29]). The open hypermedia initiative, in turn, inspired a movement towards structured computing (e.g., [1,3,31]) as an attempt to generalize the techniques and lessons learned from open hypermedia efforts.

A shared limitation of these efforts is a "heavy weight" requirement that participating tools make common use of basic utilities and structured storage. The work and the trust involved to do this has been prohibitive. Anderson [1], in reference to structured computing efforts aimed at integration, notes, for example, that an environment may require "installation of a database, .. server, … support tools … clients" and that, in general, these requirements are "too steep".

With respect to tools aimed at the end user, an irony inherent to any well-meaning effort aimed at integration is that the opposite may be the actual outcome. A new tool enthusiastically embraced

[1] http://www.w3.org/RDF/

[2] http://tools.ietf.org/html/rfc3986. See also a generalization through provision for the Universal Character Set (http://annevankesteren.nl/2005/02/iri).

for its integrative potential may later be abandoned with a gradual realization that older structures are still needed for other tools [7].

Stories of tool and system abandonment [22] point to two additional issues of "integration":

1. How to integrate into people's existing ways of working with and thinking about their information?
2. How to work with the structures and tools people already have for managing their information?

Paradoxically, people may be more inclined, eventually, to abandon existing tools and structures in favor of the new only if they are first assured that they don't have to.

1.1 The plan and purpose of this paper

This paper is structured into the following sections:

- **Planz** is described as one attempt to address the two issues of integration posed above. The section reviews the motivations, design, and current status of Planz. Most important, the section reviews lessons learned through several iterations in the development and evaluation of Planz.
- **Seven generalizations** follow from an assessment of these "lessons learned" in the prototyping of Planz. Generalizations accept a world in which there are and always will be a multitude of tools competing for our time, money and attention. Can these tools do so "non-disruptively", i.e., in ways that support the organic growth of an integrative organization of personal information over time?
- **XooML** is an XML[3]-based approach that takes a first step toward addressing these seven generalizations.

Considerable work has already gone into XooML and its proof-of-concept use in three separate tools. Even so, XooML is presented in this paper not as a completed project but rather as an effort that remains a work in progress. The paper invites the reader on a journey to explore how XooML might be used in support of a vision: Many tools; one (integrative) structure.

2 PLANZ

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.[4] The design of Planz has been guided by two principles:

1. **Organize incidentally**. Ideally, useful organizations of personal information emerge as an incidental by-product of other activities a person must do in any case. People work to complete projects. People sometimes give informal expression to a project plan, for example as a simple outline or draft document [24]. Can the structure in such a document also provide a basis for the organization of the information needed to complete the project?
2. **Organize integratively.** Integrate with existing applications and as a thin overlay to existing structures. In Planz, for example, document-like project plans are simply an alternate way to view and work with a folder hierarchy in the file system. This means that a wealth of existing tools for file management (including back-up) will still work. Planz also integrates with existing support for time and task management.[5]

2.1 On the front-end, a document

Planz displays a folder structure in either a *draft* or *outline* view (see figure 1). Planz has the affordances of a basic word processor: 1. Type directly to create or modify notes and headings. 2. Move headings and notes up or down as one might move blocks of text in an electronic document. 3. Expand and collapse headings to reveal or hide content. 4. Promote notes to be headings; demote headings to become notes.

The "document" on display in Planz can be edited to show all of a person's projects and tasks in a single, scrollable view. Headings often represent high-level projects ("Plan family vacation for summer"); subheadings then represent component tasks ("Make plane reservations"). This document also provides the basis for an organization of project- and task-relevant information via two basic operations:

- *"**Drag & link**"* to existing documents, email messages and web pages from the "outside-in". Simply select the item, drag and drop to a location within the Planz document. The item stays where it is (as a file, email message or web page) but a link pointing to this item is created within the Planz document. Or, select text as a summary or "hook" to the item. Drop the text into the Planz document plan to place a copy of the text in a new note with a link back to the item.
- ***In-context create***. Send email messages and create new documents from the "inside-out". These items are created as they would be normally (in separate windows managed by supporting applications such as the word processor or email application). However, Planz places a link to the document created or the email message sent near the insertion point within the Planz document.

The document overlay of Planz has made the realization of additional features straightforward:

- *In-place expansion*. Link to any accessible folder from within a Planz document; expand to see this folder's contents in-place.
- *Folder-focused views*. By default, Planz presents a document that is focused on a "Projects" folder.[6] However, for any folder selected in the file manager, Planz can also be invoked as a right-click option to present a view focused on this folder instead.
- *Save As HTML*. The document presented in a Planz window can be saved as an HTML file to be viewed in a Web browser or edited in a word processor. Structure as well as content is preserved.
- *Export Structure.* The structure of a Planz document can also be exported for re-use either as a project template or for immediate use in another project. For example, the folder structure of a business "trip to Boston" (with subfolders for

[3] XML, E**X**tensible **M**arkup **L**anguage, http://www.w3.org/XML/.

[4] The version of Planz (8.2) described here is a desktop application based on .NET 3.0. 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.

[5] Current integrations for time and task management are with Microsoft Outlook.

[6] "Projects" is created on installation of Planz as a sibling folder of "Documents" (or for Windows XP users, "My Projects").

"plane", "hotel", "expense voucher", etc.) might be exported for use on another business "trip to Chicago".

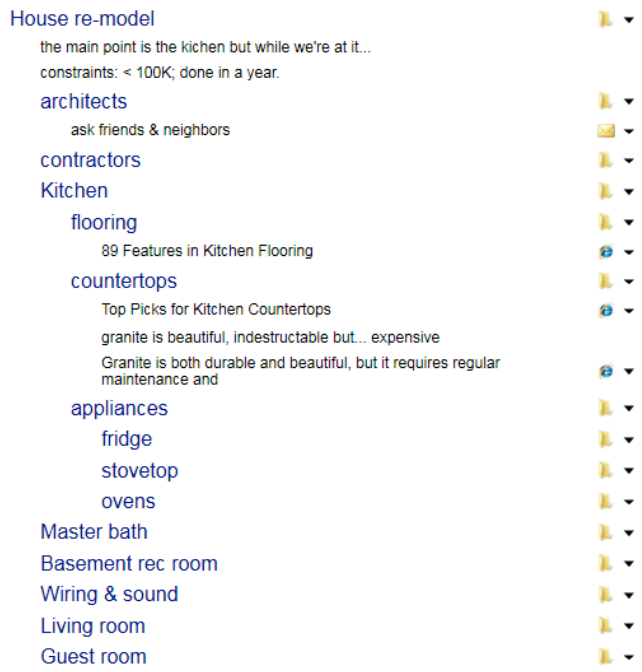

Figure 1. A sample "Plan" in Planz. Headings /subheadings represent file system folders/ subfolders. Notes can point to files, emails and web resources.

2.2 On the back-end, the file system

Headings and subheadings correspond to file system folders and subfolders. Links within Planz correspond either to local files within these folders or to shortcuts which, in turn, can point to files, web pages or email messages. The mapping from headings and links to folders and files/shortcuts is one-to-one.

In short, Planz works with the file manager to provide a new mode of file management. Users can create, modify or delete folders and files through operations initiated within Planz. In the other direction, a process of synchronization insures that document views of Planz are current with respect changes to the file system made outside Planz.

2.3 A meeting in the middle

Central to the Planz approach and its research significance, are choices made at a middle level. Planz realizes its document overlay through a dynamic, "on-demand" assembly of many small fragments of XML.

Each XML fragment defines a "Plan" – the basic unit of organization within Planz. A Plan is an ordered list of associations. An association can, but need not, have a link to an information item such as a folder, file, email or web page. An association can be "promoted" from a note to a heading which, in turn, can be expanded in-place to reveal contents. Currently this expansion works only when the information item of the expansion is a folder, but generalizations discussed later would permit this to work for other information items as well.

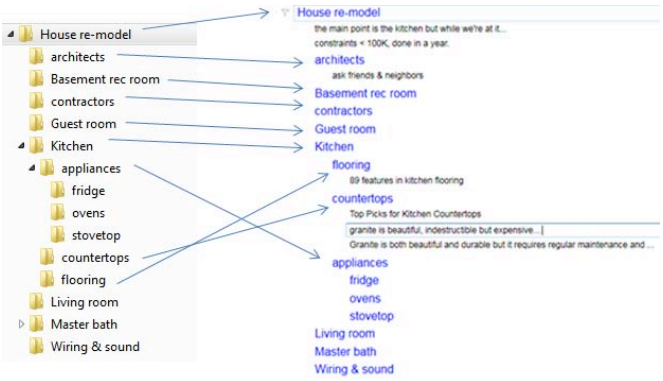

Figure 2. Planz generates a document-like view through an on-demand assembly of XML fragments, one per folder to be represented. Fragments provide additional information (e.g., informal notes and the ordering of folders as headings).

"Plans" under Planz are in one-to-one association with folders. The XML fragment defining a Plan is stored as an ordinary text file within the associated folder. The actual display of a Plan follows a process of synchronization between a Plan's XML fragment and the "reality" of the associated folder's actual contents. In cases of conflict, the file system always wins.

2.4 Current status and lessons learned

The planned development for Planz as a prototype is now complete. The code of Planz has been open-sourced[7] . Planz is available for free download[8]. Planz has a small but dedicated group of users[9] and has been the subject of numerous evaluations, informal and formal.

In earlier directive evaluations, Planz (then the Personal Project Planner) was used in a controlled setting to assess the overall approach and to prioritize feature development [25].

In a recent evaluation [26], Planz was used by eight participants on a daily basis from five to twelve days. On a measure of "project awareness" (progress and current status of a project), participant assessments gave a significant advantage to Planz over the "status quo" (a participant's existing constellation of tools and techniques). On the other hand, participant ratings also gave a significant advantage to the status quo for email management.

Comments volunteered in the concluding interview of this evaluation were generally favorable and two participants expressed the intention to continue using Planz.

The Planz that participants used was a prototype, slow to react at times and missing numerous smaller features of "fit and finish". But another impression taken from the interviews was that participants already had a "full plate" of tools for PIM. Even as participants expressed a desire – a yearning – for better, more integrative tools of PIM, they also expressed a reluctance to invest their time

[7] http://code.google.com/p/xooml/.

[8] For download: kftf.ischool.washington.edu/planner_index.htm

[9] To quote one user: "I am using Planz everyday! It is really awesome when I am trying to locate files associated with my project, links to my folders and my mail."

and information in a new tool for fear of adding further complication to their lives.

Four participants expressed appreciation for integrations with existing applications such as Microsoft Outlook and the Windows Explorer. Participants indicated that they freely switched between Planz and the Windows Explorer to take advantage of features in each. But ratings of email management suggest that other integrations were less than perfect.

What would be needed for better integration? Could integrations be Web-based and "operating-system agnostic"? What if people were able to switch freely and easily between any number of tools – picking and choosing those that best fit the current need? What if any number of tools could be brought to bear in service of a "living", growing structure of personal information?

These and related questions prompted a review of the XML-based architecture of Planz. The basic approach seemed sound: Construct large, coherent views of information from the flexible, on-demand assembly of many small XML fragments. However, the Planz of the most recent evaluation[10] made a private use of XML not formally documented.

Could a schema be designed to meet not only the needs of Planz but any number of other tools as well?

3 SEVEN GENERALIZATIONS

Experiences with Planz and the choice points of its design extend to seven directions of generalization:

G1. From some tools to many
Consider a basic folder hierarchy. This might be rendered in various ways through various tools. The Windows Explorer provides a basic tree-control. Planz supports a document-like interaction. Another tool might support a *mind-mapping* view[11]. Each interactive mode makes use of the same basic structure but each mode provides additional affordances and requires additional attributes. Using Planz, for example, people can decide to view a folder shortcut as either a note or a heading. A heading, in turn, can appear expanded or collapsed. In order for settings (heading or note, expanded or collapsed) to persist between sessions, Planz needs its own attributes. Likewise, in order to persist a mind-mapping view, a tool might need attributes to record the relative position of mind map elements. A schema should, therefore, make provision both for the basic structure and for tool-specific attribute bundles.

G2. Tool classes and metadata standards
Moving beyond a provision for individual tools, the schema might also make provision for tool classes. There might, for example, be provision for a class of mind-mapping tools or a class of "Getting Things Done" (GTD)[12] tools. Or a class of tools might be defined for a collection of items all of the same kind – songs or photographs or books. One ready way to define a class of tools is in relation to a metadata standard (for a review, see [19]) such as the Dublin Core Metadata Initiative[13]. Tools of a class would be expected to work with and provide uniform treatment for the properties defined by the standard.

G3. From file system folder to any information item
G4. The metadata fragment: from file folder to anywhere
Generalizations G3 and G4 go together. In Planz, fragments of metadata are placed in association with file folders. A generalization would provide the basis for associating fragments of metadata to other forms of information including, for example, local documents, email conversations and a variety of Web-based resources.

In Planz, a metadata fragment is actually placed as a file within the folder it describes. But what to do for other forms of information and in cases where a person lacks permission to change the item itself? A generalization, in line with G3, would support a decoupling of fragment storage from storage of the item to which a fragment applies. Fragments might, for example, be stored in a personal Web store and organized into a simple database "keyed" by the URI to which a fragment applies.

G5. The level of synchronization: From strong to varied
G6. The means of reading and modifying item structure
Generalizations G5 and G6 also go together. For file folders as represented in Planz, synchronization between fragments and the "reality" of the file folders is strong. The file manager is the primary (default) application for the management of folders and their storage. Planz is simply an alternate way of working with the same information.

An obvious but important point is that the file manager doesn't need to change nor does it need to "know" anything about Planz in order for Planz to work. All the needed interactions (to read and modify folder structure) are accomplished through an application-supported API[14].

Moving beyond the file folder to other forms of information we also need to move beyond the file manager to other primary applications and their supported APIs. In the ideal, Planz and other tools in the spirit of Planz, should be able to work with a variety of primary applications (Gmail, Evernote, Google Docs, Things, Remember The Milk, etc.) in order to read and modify the structure and content of items managed by these applications.

We also need to allow variations in API support and in the nature of the synchronization that is desired. Synchronization between metadata fragments and folder structure in Planz is strong as supported by a well-developed API.

For other forms of information, supported access may be read-only. But also, the *desired* level of synchronization may be much weaker. Metadata fragments in some cases may bear little direct resemblance to the structure or content of information items they describe. A fragment might, instead, represent "my thoughts about this item" or "things I relate to this item".

Metadata fragments may vary in their levels of synchronization to information items, in accordance with supported APIs, user permissions and also user preference. It is important, however, that

[10] This was Planz 7. The current Planz 8 has been extensively re-architected to use Windows Presentation Foundation and to use XML based on the XooML schema.

[11] Buzan, Tony. (2000). *The Mind Map Book*, Penguin Books, 1996. ISBN 978-0452273221.

[12] Allen, David (2001). *Getting Things Done: The Art of Stress-Free Productivity*. Penguin Books. ISBN 0-14-200028-0. For a review of GTD applications see LifeHaker.com's "Best of GTD", http://lifehacker.com/software/getting-things-done/best-of-gtd-161916.php.

[13] See, for example, the(http://dublincore.org/.

[14] The current version of Planz works with the Microsoft Windows Explorer and the Win32 API.

the tools work consistently in support of a given level of synchronization.

G7. From operating system-specific hierarchy to a distributed, universally accessible directed graph

The final generalization follows from the first six. Planz, as an overlay to the Windows file system, works with a hierarchy (modified to include shortcuts). In the spirit of the Web, a schema should support the platform-independent, flexible representation of a structure as a *multidigraph*[15]. A multidigraph is essentially a structure in which a node can point to another node (or to itself) via more than one link (arc, arrow). For example, Web pages and their hyperlinks define a multidigraph.

4 XOOML

As a first step towards addressing these generalizations, consider the design of an XML-based schema called *XooML* (pronounced "zoom'l") for Cross (**X**) T**o**ol **M**ark-up **L**anguage.[16]

How to develop a schema with provisions for each direction of generalization? The first direction, in particular – from one tool to many – requires special consideration. Two requirements follow:

- The schema needs to provide "growing room" for any number of tools to persist tool-specific settings as attributes.
- There must be no risk of confusion or collision between the attributes used by different tools.

We originally considered the mandatory use of namespace prefixes[17], one per tool, which could then be prepended to all attribute names used by a given tool (e.g., "thisTool" as a prefix for "thisAttribute"). But clearly, the mandatory use of prefixes carries its own issues of confusion and potential collision. For example, we might select the prefix "planz" for the Planz tool. But suppose another tool decided to use this prefix as well? Checking all XooML fragments for potential conflicts isn't feasible. But neither is the maintenance of a central registry of "Prefixes used by tools that use XooML".

A better approach bundles 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[18]. The use of elements to bundle attributes happens at two levels:

1. For any given information item, the XooML schema defines the structure of an associated fragment of metadata. As depicted in Figure 3, a fragment is a bundling of fragment-level "common" (tool-independent) attributes followed in sequence by zero or more bundles of tool-specific attributes (*fragmentToolAttributes*) followed by zero or more elements of type *association*.

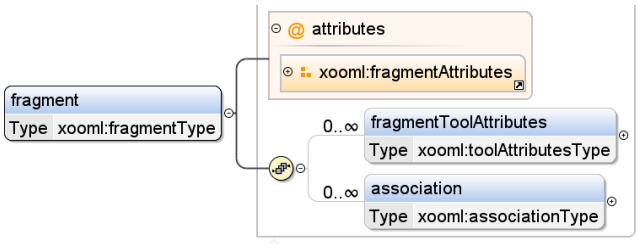

Figure 3. A XooML fragment is a bundling of "common" (tool-independent) attributes followed by zero or more bundles of tool-specific attributes (*fragmentToolAttributes*) followed by zero or more *associations*.

2. This pattern is partially repeated for each association element within a fragment. As depicted in Figure 4, an association is a bundling of common (tool-independent attributes) followed in sequence by zero or more bundles of tool-specific attributes.

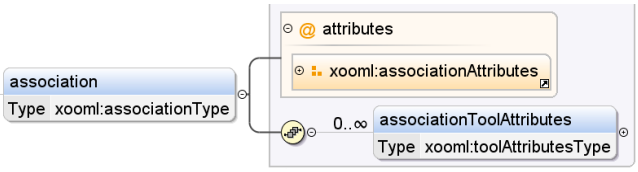

Figure 4. An association is a bundling of common (tool-independent attributes) followed by zero or more bundles of tool-specific attributes.

Essentially, each XooML fragment is bundling information for a node and its outgoing links. Fragments in aggregate define a (multi)digraph.

```
<?xml version="1.0" encoding="UTF-8"?>
<xooml:fragment xmlns:xooml="http://kftf.ischool.washington.edu/xmlns/xooml"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://kftf.ischool.washington.edu/xmlns/xooml  http:/
    schemaVersion="0.42"
    relatedItem="file:C:\Users\johnj\Projects"
    defaultApplication="file:%windir%\explorer.exe"
    levelOfSynchronization="strong">
    <xooml:fragmentToolAttributes xmlns="http://kftf.ischool.washington.edu/xm
        toolVersion="8.2"
        toolName="Planz"
        showAssociationsMarkedDone=FALSE
        showAssociationsMarkedDefer=FALSE/>
    <xooml:association ID="7bbd9780-a8bd-4114-b2bb-647a33e89c47"
        associatedItem="http://kftf.ischool.washington.edu/planner_index.htm"
        associatedIcon=""
        associatedXooMLFragment=""
        displayText="When there's time, try out Planz..."
        openWithDefault=""
        <xooml:associationToolAttributes xmlns="http://kftf.ischool.washingtor
            isVisible=TRUE
            isHeading=FALSE
            isCollapsed=FALSE
```

Figure 5. A snippet of a XooML fragment.

A snippet from a XooML fragment is displayed in Figure 5. A XooML fragment represents attributes at two levels – for the fragment and for each of its associations. At each level, some attributes are tool-independent or "common"; other attributes are tool-specific. Consequently, the attributes of a fragment are readily summarized in a matrix (see The XooML schema is currently being actively used by three separate tools:

- ***Planz*** (version 8.2) as discussed above.

[15] See, for example, Bollobas, Bela; *Modern Graph Theory*, Springer; 1st edition (August 12, 2002). ISBN 0-387-98488-7.

[16] See kftf.ischool.washington.edu/XMLschema/0.41/XooML.xsd

[17] http://www.w3.org/TR/REC-xml-names/.

[18] E.g., xmlns=http://kftf.ischool.washington.edu/xmlns/planz. To aid in readability tool might still include a tool-appropriate prefix in its namespace declarations and then prepend this prefix to its attributes. But doing so is optional.

- ***QuickCapture*** – invoked with a click of Windows (the flag key) + c and used to capture a link to the item (document, email message, web page) in the 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*** – created by "wrapping" the open-source FreeMind tool with support for XooML.

Table 1).

The XooML schema is currently being actively used by three separate tools:

- ***Planz*** (version 8.2) as discussed above.
- ***QuickCapture***[19] – invoked with a click of Windows (the flag key) + c and used to capture a link to the item (document, email message, web page) in the 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*** – created by "wrapping" the open-source FreeMind[20] tool with support for XooML.[21]

Table 1. A matrix summary of fragment attributes. (Values are mandatory only for attributes in bold).

	Fragment	Association
Common[22]	***schemaVersion*** ***relatedItem*** *defaultApplication* *levelOfSynchronization* …	***ID*** *associatedItem* *associatedIcon* *associatedXooMLFragment* *displayText* *openWithDefault* …
Tool-specific (Planz)	*toolVersion* *toolName* *showAssociationsMarkedDone* *showAssociationsMarkedDefer* ….	*isVisible* *is Heading* *isCollapsed* …
Tool-specific (FreeMind)	*toolVersion* *toolName* …	*radius* *polarAngle*
		…
Another tool...	…	…

[19] QuickCapture comes with the installation of Planz and can also be installed separately. QuickCapture was developed in the spring of 2010 by our assistant developer, Zhyong (Joe) Xie.

[20] http://freemind.sourceforge.net/wiki/index.php/Main_Page

[21] This wrapper was created by our lead developer, Eric Sheng Bi, in the spring of 2010.

[22] Other common attributes include *createdBy (URI), createdOn (time/date), lastModifiedBy* and *lastModifiedOn.*

FreeMindX, directed to the same "House re-model" depicted in Figure 1, yields the mind-map view shown in Figure 6. "xooml.xml" fragments have shared use by each tool and can be easily inspected to see how XooML works in practice.

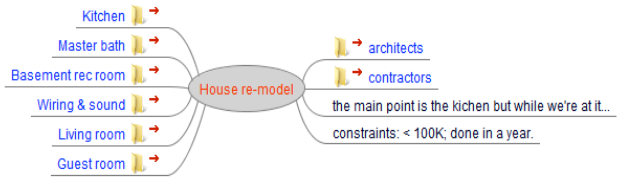

Figure 6. The house re-model of Figure 1 rendered as a mind map in FreeMindX.

4.1 XooML for seven generalizations

The XooML schema provides support for each of the seven generalizations:

G1. From some tools to many

Any tool can push its attributes into a XooML fragment as a way to add metadata both for the fragment overall (and its related item) and for each of a fragment's associations. Tool-specific attributes are bundled into sub-elements in a fragment. Sub-elements are identified through a namespace declaration (*xmlns*).

For example, Planz uses the fragment-level attributes of *showAssociationsMarkedDone* and *showAssociationsMarkedDefer* (Table 1) to determine whether to display associations that have been deferred or marked as "done". Planz uses association-level attributes to determine whether or not to display an association (*isVisible)* and, if so, as a heading or a note (*isHeading)* and, if a heading, then expanded or collapsed (*isCollapsed)*. Likewise, a mind-mapping tool such as FreeMind might use attributes to determine the relative position of one node to another (e.g., *radius, polarAngle)*. Tools can do what they like, so to speak, within their sub-elements. Attributes needn't be defined or named according to (any) convention as long, of course, as the tool itself can make good use of attribute information.

G2. Tool classes and metadata standards

The namespace declaration of a sub-element might, instead, refer to a metadata standard. Tools reading and writing to sub-elements so identified would be expected to "understand" and consistently use the attributes defined through the standard. Tools working with Dublin Core metadata would, for example, be expected to make consistent use of attributes such as "Title", "Creator" and "Publisher".

G3 thru G6

Generalizations G3 thru G6 are in direct correspondence to XooML attributes (With respect to G6, the means of enumerating and modifying item structure, simply pointing to a *defaultApplica-*

tion is only one step. Also needed is an appropriate interface through which to enumerate and modify item structure.

Table 2). However, correspondence is only the first step. Questions remain, especially in relation to G5 and G6.

With respect to G6, the means of enumerating and modifying item structure, simply pointing to a *defaultApplication* is only one step. Also needed is an appropriate interface through which to enumerate and modify item structure.[23]

Table 2. Generalizations and XooML attributes

G3	From file folder to any information item	***relatedItem,*** *associatedItem*
G4	The XML fragment: from file system folder to anywhere	*Associated-XooMLFragment*
G5	The level of synchronization	*levelOfSynchronization*
G6	Means of reading and modifying item structure	*defaultApplication*

Questions also arise concerning the address values in ***relatedItem,*** *associatedItem* and associatedXooMLFragment. In most cases, the value is an absolute (fully qualified) URI. However, in some cases, the value may be a partial address that combines with a base address to yield an absolute URI. In Planz, for example, the *associatedItem* is given a relative path when pointing to a "local" file or subfolder. An absolute path is derived by concatenating this path to the path for the parent folder. Addresses then remain valid even as parent folders are moved within the folder hierarchy.

G7. From operating system-specific hierarchy to a distributed, universally accessible directed graph

Generalizations combine to support a structure that is, potentially, highly distributed. XooML fragments might be local to a person's personal file system or up on the Web. Fragments might be placed inside the information item they describe – as a file inside a folder, for example, or possibly even as a private stream of text inside a document. Or fragments might be placed in "nooks and crannies" of space freely available on the Web. Large groups of fragments might be organized into a simple, central, Web-based relational database.[24]

Each fragment, as a metadata description of an information item, is, in its own right, a grouping of information. Its associations define a unit of choice – the subfolders and files to open under a folder, for example, or the hyperlinks to click from a Web page.

More important, fragments link together via their associations create a thin integrative overlay that "floats" over the information items they describe. A vision is that people might create, modify and use this structure by picking and choosing from a large and growing arsenal of XooML-supporting tools.

[23] Something similar, for example, to the **IshellFolder** interface and its support for an *EnumObjects* method (http://msdn.microsoft.com/en-us/library/bb775075%28VS.85%29.aspx).

[24] For example, a SQLite database (http://www.sqlite.org/) with records keyed by *relatedItem.*

4.2 Working with XooML

Fragments are used by a XooML-supporting tool such as Planz (8.0) or FreeMindX to generate a coherent view through the following steps:

1. **Process an initial fragment** and synchronize its metadata with the *relatedItem*. In Planz, the initial fragment, by default, is the XooML fragment for a "Projects" folder. The contents of this fragment are compared with the actual contents of the "Projects" folder. In cases, of conflict, the fragment is modified to agree with the file system. From this top-level synchronized XooML fragment, Planz builds a top-level *Plan.*
2. **Recursively retrieve and process additional XooML fragments as needed.** In Planz, for example, the subfolders and folder shortcuts of a folder appear within the folder's Plan as document-like *heading* associations. For each of these headings that were last shown as "expanded", Planz retrieves folder information and an associated XooML fragment and then uses the results of their synchronization to determine the display of sub-Plans.
3. **View completion is tool-dependent**. In Planz, the process completes when sub-plans have been generated for each in a list of "expanded" associations encountered during the processing of XooML fragments. In another tool, display generation might stop when some number of fragments has already been displayed or when fragments have been displayed to a certain depth from a starting fragment.

This process works for a wide variety of what might be call tree-view tools – tools designed to display and work with hierarchically structured information. Hierarchies are widely used in the organization of personal information [15]. For purposes of visualization and manipulation, hierarchies are often represented as trees – i.e., as connected, acyclic graphs.

Herman et al [20] review a large number of distinct tree visualizations including H-tree layout, radial view, balloon view, and treemap. Several forms of tree visualization are found in file managers such as the Macintosh Finder and Microsoft's Windows Explorer.[25] Mind maps, in all their variety,[26] including those of the PersonalBrain [27], are tree views.

Farther afield, are views of a conventional hierarchically structured document (e.g., with headings and subheadings marking different levels). Even Web pages provide a kind of tree view since most conform to the hierarchical Cascading Style Sheets (*CSS*) box model[28].

All tree views, broadly defined in this way, can follow the same three steps of rendering listed above. For example, a hypothetical "box-model" tool wrapped with XooML support might "pour" a portion of the directed graph defined by XooML fragments into a newly generated Web page. Associations in a fragment would be represented as sub-boxes (until some stopping function is reached). If the initial tool-generated layout of boxes and sub-boxes is modified by the user, then this layout can persist in XooML through association-level, tool-specific attributes.

[25] For a review see, http://en.wikipedia.org/wiki/File_manager.

[26] For a sampling, http://en.wikipedia.org/wiki/Mind_map.

[27] http://www.thebrain.com/.

[28] http://www.w3.org/TR/CSS2/box.html.

The node represented by a XooML fragment can be bushy – it can have hundreds or thousands of associations to represent, for example, a collection of photographs, songs, articles, etc. Items in such a collection are often characterized by and distinguished from each other by a common set of properties (e.g., author, title, year of publication).In such cases, fragment-level, tool-specific attributes might be used to store the values needed to re-create the current view (e.g. properties on display, sort order, etc.).

Or consider the somewhat curious case of Microsoft OneNote. A XooML structure could be "poured" into the standard OneNote view in a way that takes advantage of OneNote's provision for section tabs (on top) and page tabs (to the right side):

1. Use a starting XooML fragment to fill the first page of a section tab. Each association becomes a separate text box on the page with layout initially determined by OneNote.
2. Fill additional pages, each with an *associatedXooMLFragment reached* from an association of this "front-page" fragment.

Again, as with the box-model tool, if the initial layout of boxes and sub-boxes is modified by the user, this layout can persist in XooML through association-level, tool-specific attributes.

4.3 Related Work

XML – Extensible Markup Language – lives up to its name. Its extensibility has been used, as intended, to define a wide-variety of sub-languages[29]. XooML is one. Additional standards of representation and interchange are generated by the Semantic Web initiative[30] and "sub-initiatives" such as Linked Data[31].

XooML relates to several of these efforts. First, the seven generalizations might have been supported using **RDF** rather than XML. Or generalizations might have been supported using **JSON** (JavaScript Object Notation)[32], a standard for data interchange.

The choice of XML was partly a matter of available expertise[33] and partly due to XML's focus on the generalized notion of a document. More specifically, in contrast to both RDF and JSON, the means for declaration of an XML schema provide more straightforward support for the representation of document structure [21].

The structure specified in XooML is minimal but essential. The fragment must often serve as an indivisible grouping of information, establishing a context for the comprehension of and selection among its constituent associations. Many properties – "above", "to the right of" or "most recent among" – are emergent from and only make sense at the level of this grouping. The fragment is a molecule, so to speak, to the atoms of it constituent associations.

Also critical in the XooML fragment and its grouping of associations are attributes needed to establish information provenance. Who (which tool) created a fragment or an individual association? When? Which other tools have modified same? When? An XML schema can make explicit provision for these and other attributes (as these prove necessary). A fragment can be determined to be invalid if these attributes are not present.

On the other hand, RDF statements can be generated directly from the information in a XooML fragment[34]. For example, the ***relatedItem*** of the XooML fragment can be the subject of well-formed statements involving, as predicates and objects, the attributes and values from both the "common" XooML namespace and from tool-specific (metadata-specific) sub-elements. Statements also follow directly from the associations of a XooML fragment.

In its multi-tool focus, XooML also compares to several initiatives that use XML to give uniform tool-independent expression to the user interface (UI). These include **UsiXML** (User Interface eXtensible Markup Language)[35], **XUL** (User Interface Language)[36], and **UIML** (User Interface Markup Language)[37]. These XML-based sub-languages can be used to express a variety of UI constructs from simple messages (e.g., "Hello World!") to sophisticated compound menus.

Initiatives towards tool-independent representation of the UI are perfectly complementary to XooML. XooML aims towards a diversity of expressions of and interactions with the *same* structure across a variety of tools. The user interface initiatives listed above aim towards a uniform representation of and (reasonably) consistent expression of the "same" UI constructs across observant tools. The aim at a lower level might be, for example, that the primary menu toolbar looks and behaves similarly across tools. The aim at a higher level might be for all mind-mapping applications to behave consistently with each other. The higher-level aim may be at cross-purposes to an aim to compete and the lower-level aim may be better achieved through other means (e.g., shared components). Regardless, these aims are orthogonal to the one structure/many tools vision of XooML.

Also of relevance but independent of XooML are stylesheet languages designed to promote a consistent appearance of documents (with more general application to the UI and the user experience). These include **CSS** (Cascading Style Sheets) [38] and **XSL** (Extensible Stylesheet Language)[39] **formatting objects**.

For example, Planz uses a rudimentary style sheet to enable and record user-initiated changes in font size for headings and notes. These settings are stored separately from XooML. Clearly, it defeats the "normalizing" purpose of a style sheet if its settings are stored repeatedly across XooML fragments and their associations. More generally, the use of style sheets is completely up to the individual tool and has nothing directly to do with XooML.

5 DISCUSSION

The XooML schema defines a flexible, extensible sub-language of XML for use to represent what is essentially a directed graph of

[29] See://en.wikipedia.org/wiki/List_of_XML_markup_languages.

[30] http://semanticweb.org/wiki/Main_Page.

[31] http://linkeddata.org/.

[32] http://www.json.org/.

[33] We were fortunate to have the involvement of Jasper Bleijs as our XML expert.

[34] RDF/XML (http://www.w3.org/TR/REC-rdf-syntax/) as one XML-based serialization of an RDF graph can be generated automatically from XooML via XSLT (Extensible Stylesheet Language Transformations, http://www.w3.org/TR/xslt20/).

[35] http://www.usixml.org/

[36] https://developer.mozilla.org/En/XUL.

[37] http://www.uiml.org/specs/index.htm.

[38] http://www.w3.org/Style/CSS/.

[39] http://www.w3.org/TR/xsl11/.

nodes pointing to other nodes. A node, as described by a XooML fragment, can represent any item that is addressable by a URI including folders, tags, email messages, local documents, and Web pages.

A fragment is metadata for its *relatedItem.* This metadata may closely adhere to the item's content and structure or not. The metadata may, instead for example, represent the user's thoughts about an item ("my review of …") or include links to other items that "I want to access while I'm working on this document".

Fragments can combine to support alternate modes of working with existing structures, such as folder or tag hierarchies or collections of songs, photos and articles. Fragments can combine to define brand-new structures as well. Fragments can "live" anywhere, from the local file system to a database on the Web.

Most important, XooML fragments and their associations can include any number of new attributes in support of a specific tool or a class of tools. One structure, supported by many tools.

5.1 Variations in the use of XooML

XooML allows for several variations in its use. Space allows for the mention of only a few:

5.1.1 Piggy-back use of someone else's taxonomy

Why develop a brand-new organization when others may be available already? Personally owned XooML fragments might be placed in association with the categories of a publicly available, Web-based taxonomy. Taxonomies (categories and classification schemes) are out there for the choosing ranging from the Library of Congress Classification Outline[40] to the categories used by Stephen Covey on his blog.[41]

5.1.2 Groupwork

XooML readily scales to a small team situation. A detailed discussion of considerations involved (e.g., how conflicts are detected and resolved) is beyond the scope of this paper. However, two aspects of XooML are worthy of mention:

1. The granularity of XooML. XooML fragments are modular and usually small. Even though two or more people are working on the same "document" (e.g. a larger structure as displayed in a tool like Planz or FreeMind), the chances that people are trying to change the same fragment at the same time are small.
2. The granularity of committed modifications. XooML admits to a fine-grained, even character-by-character, commit of modifications. This means, in turn, that a failure of commitment – e.g., in a case where the XooML fragment has been modified by someone else since the user's current view was generated – results, in worst case, in only a small loss of time and data.

5.1.3 XooML fragments as items in their own right

A XooML fragment is addressable; it has a URI. A XooML fragment might itself become a *relatedItem* that is the subject of the metadata of another XooML fragment. This might happen, for example, if John wishes to subscribe to XooML annotations made by Jill, where these annotations, in turn, might have a Web page as a *relatedItem.*

[40] http://www.loc.gov/catdir/cpso/lcco/.

[41] https://www.stephencovey.com/blog/.

5.2 Next steps in XooML R&D

Next steps include:

1. A toolkit to support consistent use of XooML through a provision for basic manipulations of XooML fragments (e.g., getAttributeValue, setAttributeValue, createAssociation, destroyAssociation, …)
2. A log for recording informational events. The log can be read from and written to directly by participating tools and is also written to as an incidental by-product of XooML toolkit use.
3. The development of XooML wrappers for more OSS tools.[42]
4. User studies aimed at understanding when the re-purposing of structure through different tools is helpful and when it isn't. Will people recognize the common structure presented, in very different ways, by different tools?

6 CONCLUSION

Efforts on XooML reach towards a new age in information management wherein organizing information structures grow and change over time as driven by the internal needs of their owners and not the external demands of tools. People may eventually be empowered to create personal unifying taxonomic structures (PUTs) [23, Chapter 14] to organize all information of personal relevance into one, coherent, sustainable, distributed structure.

XooML is described in this paper not as a final word but rather as an illustration of an approach that accepts a world in which:

- There will always be a diversity of tools competing for our time and attention.
- An integration of personal information is not likely to happen through some new release of a desktop operating system nor through a Web-based "super tool".
- Instead, integration may best be supported through the development of a standards-based infrastructure that makes provision for the shared manipulation of common structure by any number of tools, each in its own way.

XooML itself is quite simple. XooML provides for a kind of passive structured "scratchpad" through which tools communicate. The XooML schema defines the structure for a fragment of metadata which is placed in association with an information item such as a folder, document, email message, web page or, more generally still, anything "of interest" that can be addressed by a URI.

A key aspect of schema structure is a factorization of "common" from "tool-specific" attributes. Factorization is done both at the level of a metadata fragment and each of its associations. Essentially, a fragment is a collection of attribute bundles where bundles are identified through a namespace declaration. By the "rules"[43] any tool can inspect and modify not only the attributes in its bundles but also the attributes in common bundles. A tool can inspect but should not modify the attributes of another tool.

[42] There are many to choose from (e.g., http://en.wikipedia.org/wiki/Commercial_open_source_applications#List_of_Commercial_Open_Source_Applications_and_Services)

[43] Rules are expected to be respected by participating tools but are not enforced.

When tools work interchangeably with the same structure, the costs to try a new tool may be much reduced. Tools can work independently but in aggregate to support a much greater diversity in the needs of people as these change over time and with the task at hand.

7 ACKNOWLEDGMENTS

This material is based on work supported by the National Science Foundation (#0534386). Many thanks to Jonathan Grudin, Ken Anderson, Joe Tennis and two anonymous reviewers who provided useful feedback on earlier drafts of this paper.

8 REFERENCES

[1] 1. Anderson, K.M. Towards lightweight structural computing techniques with the SmallSC framework. *Proceedings of the 2005 symposia on Metainformatics*, ACM (2005), 1.

[2] 2. Anderson, K.M., Sherba, S.A., and Lepthien, W.V. Towards large-scale information integration. *Proceedings of the 24th International Conference on Software Engineering*, ACM (2002), 524-534.

[3] 3. Anderson, K.M., Sherba, S.A., and Lepthien, W.V. Structure and behavior awareness in themis. *Proceedings of the fourteenth ACM conference on Hypertext and hypermedia*, ACM (2003), 138-147.

[4] 4. Anderson, K.M., Taylor, R.N., and E. James Whitehead, J. Chimera: hypertext for heterogeneous software environments. *Proceedings of the 1994 ACM European conference on Hypermedia technology*, ACM (1994), 94-107.

[5] 5. Bergman, O., Beyth-Marom, R., and Nachmias, R. The project fragmentation problem in personal information management. *Proceedings of the SIGCHI conference on Human Factors in computing systems*, ACM (2006), 271-274.

[6] 6. Boardman, R. and Sasse, M.A. "Stuff goes into the computer and doesn't come out" A cross-tool study of personal information management. *ACM SIGCHI Conference on Human Factors in Computing Systems (CHI 2004)*, (2004).

[7] 7. Bruce, H., Jones, W., and Dumais, S. Information behavior that keeps found things found. *Information Research 10*, 1 (2004).

[8] 8. Bruce, H., Wenning, A., Jones, E., Vinson, J., and Jones, W. Seeking an ideal solution to the management of personal information collections. *Information Seeking in Context Conference -(ISIC) 2010*, (2010).

[9] 9. Bush, V. As We May Think. *The Atlantic Monthly 176*, 1945, 641-649.

[10] 10. Catarci, T., Dong, L., Halevy, A., and Poggi, A. Structure everything. In *Personal Information Management*. University of Washington Press, 2007.

[11] 11. Chaffee, J. and Gauch, S. Personal ontologies for web navigation. *CIKM 2000 : 9th International Conference on Information Knowledge Management*, ACM (2000), 227-234.

[12] 12. Cruz, I.F. and Xiao, H. A layered framework supporting personal information integration and application design for the semantic desktop. *The VLDB Journal 17*, 6 (2008), 1385-1406.

[13] 13. Davis, H., Hall, W., Heath, I., Hill, G., and Wilkins, R. Towards an integrated information environment with open hypermedia systems. *Proceedings of the ACM conference on Hypertext*, ACM (1992), 181-190.

[14] 14. Dourish, P., Edwards, W., LaMarca, A., and Salisbury, M. Presto: an experimental architecture for fluid interactive document spaces. *ACM Transactions on Computer-Human Interaction 6*, 2 (1999), 133-161.

[15] 15. Dourish, P., Edwards, W., LaMarca, A., and Salisbury, M. Using properties for uniform interaction in the Presto Document System. *The 12th Annual ACM Symposium on User Interface Software and Technology (UIST'99)*, (1999).

[16] 16. Freeman, E. and Gelernter, D. Lifestreams: A storage model for personal data. *ACM SIGMOD Record (ACM Special Interest Group on Management of Data) 25*, 1 (1996), 80-86.

[17] 17. Geambasu, R., Balazinska, M., Gribble, S.D., and Levy, H.M. Homeviews: peer-to-peer middleware for personal data sharing applications. *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, ACM (2007), 235-246.

[18] 18. Gemmell, J., Bell, G., and Lueder, R. MyLifeBits: a personal database for everything. *Commun. ACM 49*, 1 (2006), 88-95.

[19] 19. Greenberg. Metadata and Digital Information. In *Encyclopedia of Library and Information*. 2010, 3610 — 3623.

[20] 20. Herman, I., Melançon, G., and Marshall, M.S. Graph Visualization and Navigation in Information Visualization: A Survey. *IEEE Transactions on Visualization and Computer Graphics 6*, 1 (2000), 24-43.

[21] 21. Hunter, J. and Lagoze, C. Combining RDF and XML schemas to enhance interoperability between metadata application profiles. *Proceedings of the 10th international conference on World Wide Web*, ACM (2001), 457-466.

[22] 22. Jones, E., Bruce, H., Klasnja, P., and Jones, W. "I Give Up!" Five Factors that Contribute to the Abandonment of Information Management Strategies. (2008).

[23] 23. Jones, W. *Keeping Found Things Found: The Study and Practice of Personal Information Management*. Morgan Kaufmann Publishers, San Francisco, CA, 2007.

[24] 24. Jones, W., Bruce, H., Foxley, A., and Munat, C. Planning personal projects and organizing personal information. *69th Annual Meeting of the American Society for Information Science and Technology (ASIST 2006)*, American Society for Information Science & Technology (2006).

[25] 25. Jones, W., Klasnja, P., Civan, A., and Adcock, M. The Personal Project Planner: Planning to Organize Personal Information. *ACM SIGCHI Conference on Human Factors in Computing Systems (CHI 2008)*, ACM, New York, NY (2008), 681-684.

[26] 26. Jones, W., Hou, D., Sethanandha, B.D., Bi, S., and Gemmell, J. Planz to put our digital information in its place. *Proceedings of the 28th of the international conference extended abstracts on Human factors in computing systems*, ACM (2010), 2803-2812.

[27] 27. Karger, D.R. Unify Everything: It's All the Same to Me. In *Personal Information Management.* University of Washington Press, Seattle, WA, 2007.

[28] 28. Karger, D.R., Bakshi, K., Huynh, D., Quan, D., and Sinha, V. Haystack: A general purpose information management tool for end users of semistructured data. *Second Biennial Conference on Innovative Data Systems Research (CIDR 2005)*, (2005).

[29] 29. Karousos, N., Pandis, I., Reich, S., and Tzagarakis, M. Offering open hypermedia services to the WWW: a step-by-step approach for developers. *Proceedings of the 12th international conference on World Wide Web*, ACM (2003), 482-489.

[30] 30. Katifori, A., Halatsis, C., Lepouras, G., Vassilakis, C., and Giannopoulou, E. Ontology visualization methods\—a survey. *ACM Comput. Surv. 39*, 4 (2007), 10.

[31] 31. Nurnberg, P.J., Wiil, U.K., and Hicks, D.L. Rethinking structural computing infrastructures. *Proceedings of the fifteenth ACM conference on Hypertext and hypermedia*, ACM (2004), 239-246.

[32] 32. Ravasio, P., Schär, S.G., and Krueger, H. In pursuit of desktop evolution: User problems and practices with modern desktop systems. *ACM Trans. Comput.-Hum. Interact. 11*, 2 (2004), 156-180.