VikiBuilder: End-user Specification and Generation of Visual Wikis

Christian Hirsch, John Hosking
Department of Computer Science
The University of Auckland
Auckland, New Zealand
chir008@aucklanduni.ac.nz,
j.hosking@auckland.ac.nz

John Grundy
Faculty of Information & Communication Technologies
Swinburne University of Technology
Melbourne, Australia
jgrundy@swin.edu.au

ABSTRACT
With the need to make sense out of large and constantly growing information spaces, tools to support information management are becoming increasingly valuable. In prior work we proposed the "Visual Wiki" concept to describe and implement web-based information management applications. By focusing on the integration of two promising approaches, visualizations and collaboration tools, our Visual Wiki work explored synergies and demonstrated the value of the concept. Building on this, we introduce "VikiBuilder", a Visual Wiki meta-tool, which provides end-user supported modeling and automatic generation of Visual Wiki instances. We describe the design and implementation of the VikiBuilder including its architecture, a domain specific visual language for modeling Visual Wikis, and automatic generation of those. To demonstrate the utility of the tool, we have used it to construct a variety of different Visual Wikis. We describe the construction of Visual Wikis and discuss the strengths and weaknesses of our meta-tool approach.

Categories and Subject Descriptors
D.2 [**Software Architectures**]: Software Architectures; H.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; H.5 [**Information Interfaces and Presentation**]: Hypertext/Hypermedia.

General Terms
Design, Documentation, Human Factors.

Keywords
Visual Wiki, knowledge management, visualization, code generation, domain specific visual language, modeling.

1. INTRODUCTION
Wikis have become increasingly popular for collaboratively creating, managing and sharing knowledge. This includes both for large scale, general purpose knowledge repositories, such as Wikipedia, but also increasingly for corporate usage [15]. Wikis have significant advantages in their open collaborative approach to (often tacit) knowledge capture and manipulation (wiki "gardening"). However, the popularity of wikis does lead to a problem with scale: as the size of wikis increase, users increasingly find it difficult to locate and assimilate the knowledge they need [3]. As a result, while establishing a "wiki culture" within an organization is readily achievable, getting people to sustain their usage of a wiki can be problematic due to these search and navigation issues. To mitigate these issues, we have been exploring the concept of a Visual Wiki [13]. This combines visualizations, providing a high level overview, and wiki pages, providing more detailed information juxtaposed in a focus-plus-context oriented format. Having successfully manually developed and evaluated a variety of applications based on our Visual Wiki conceptual model, we wanted to make the development of Visual Wiki style applications easier, ideally by end-users themselves. Accordingly we have designed and implemented VikiBuilder, a meta-tool supporting the specification and realization of Visual Wiki applications. We begin by motivating our research, including an introduction to our Visual Wiki conceptual model. We then provide a high level description of our VikiBuilder approach and the meta-tool realizing it. Following a more detailed description of the application's architecture and implementation, we prove its utility by describing its use to implement a variety of Visual Wikis. This leads into discussion of our experiences using and evaluating VikiBuilder and we outline some areas for future research.

ASE'10, September 20–24, 2010, Antwerp, Belgium.
Copyright 2010 ACM 978-1-4503-0116-9/10/09...$10.00.

2. MOTIVATION
A Visual Wiki [13] is a web-application integrating a textual and a visual representation of the same underlying body of knowledge. Both or either of the representations may be editable in a shared, traditional wiki style. The purpose of a Visual Wiki is to increase the effectiveness of wikis as knowledge management tools, via visual enhancements. As shown in Figure 1 our Visual Wiki concept consists of four components: the problem domain, the textual and visual representation, and a mapping in between.

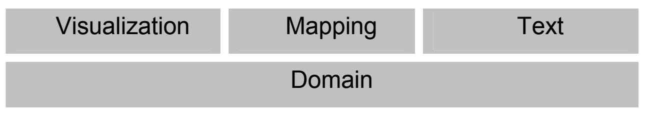


Figure 1. The four components of the Visual Wiki.

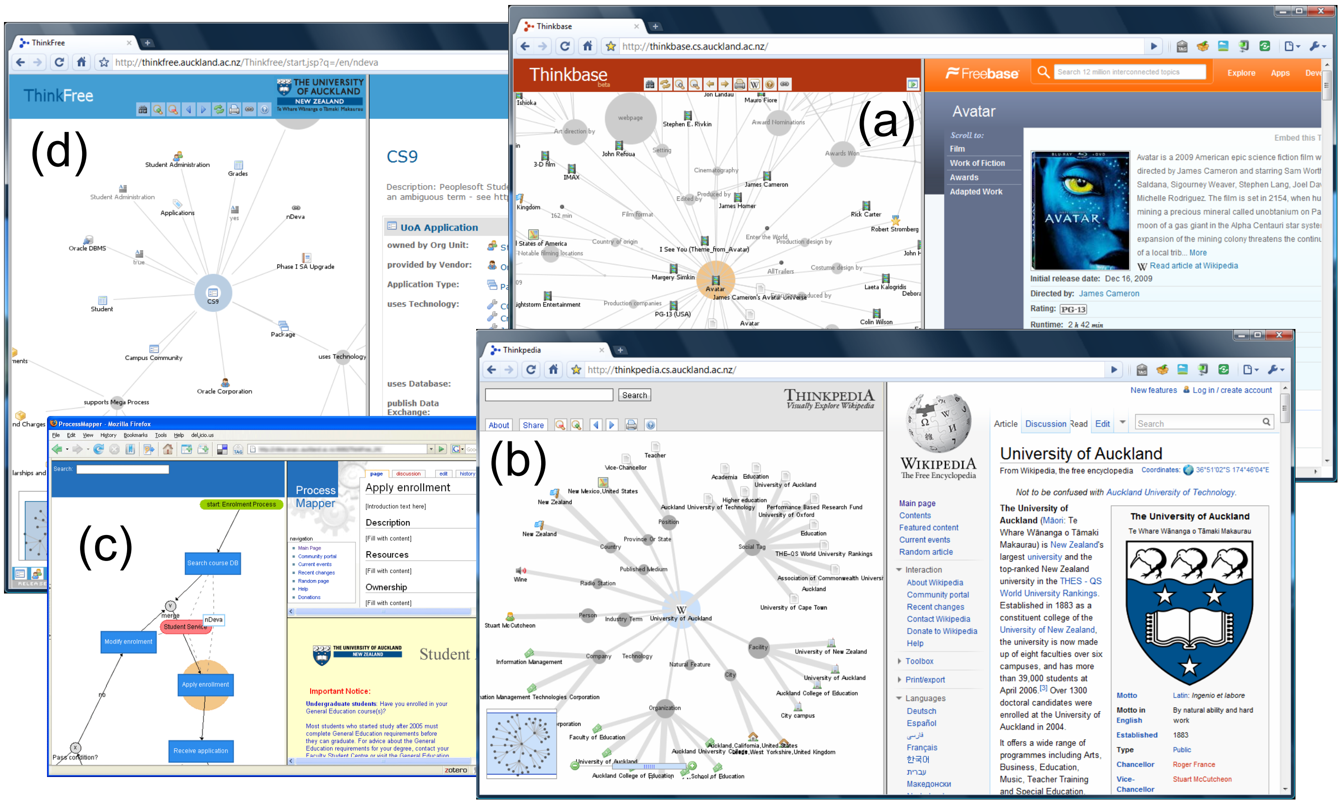


Figure 2. Visual wiki applications: (a) Thinkbase (b) Thinkpedia (c) ProcessMapper (d) ThinkFree.

The *domain* component describes the purpose and content of the Visual Wiki. For example, it could be meant for tasks such as search and exploration, or creation of information. The content specifies the problem domain information the application supports. The *text* and *visualization* components are similar and provide textual and visual interfaces to surface and interact with the domain content. The *mapping* component determines how the two representations are coordinated. This includes both the navigation behavior and consistency management policies. A more thorough discussion of our Visual Wiki concept can be found in [13].

Figure 2 shows several of our previously developed Visual Wiki applications. *Thinkbase*[1] (a) [11-13] is a visual navigation tool for Freebase[2], an open semantic wiki. The frame at the left shows a force-directed interactive graph of the relationships between the currently selected Freebase page (on the movie *Avatar*) and other semantic entities (nodes). Different icons represent different types of entities, aggregation nodes (in grey) collect entities related in a similar semantic way. Nodes can be expanded or collapsed. The frame on the right displays the Freebase wiki page for the currently selected topic. Users can navigate and explore the information space via the visualization. Additional search capabilities, accessible through context menus, are provided. We have also adapted Thinkbase for use in modeling and displaying software architecture documentation in *KaitoroBase* [24]. *Thinkpedia*[3] (b) [12] provides similar functionality for Wikipedia. As Wikipedia is less structured, we have used the SemanticProxy[4] web service to “semantify” the wiki page content to produce the relationship graph. The width of edges specifies the strength of the semantic relationship as evaluated by the SemanticProxy. *ProcessMapper* (c) [13] visualizes business processes specified using BPMN and coordinates the visualization with wiki pages documenting process stages and organizational information and web applications realizing them. *ThinkFree* (d) is a Visual Wiki, deployed as an enterprise application, describing Enterprise IT assets at the University of Auckland. The asset descriptions are maintained in Freebase, but the visualization coordinates both, these descriptions as well as corporate wiki pages (in Confluence) and SharePoint documents relevant to an asset.

These applications, their evaluation and corporate deployment (in case of ThinkFree), have together demonstrated that the Visual Wiki concept provides considerable value for knowledge management [12, 13]. However, constructing each of the applications involves significant programming. While we have leveraged industrial strength components, such as the Thinkmap[5] visualization toolkit, to reduce development, each application has still required substantial programming efforts.

[1] http://thinkbase.cs.auckland.ac.nz
[2] http://www.freebase.com
[3] http://thinkpedia.cs.auckland.ac.nz
[4] http://semanticproxy.com
[5] http://www.thinkmap.com/

Many applications deal with visualizations of some kind of data or information and the display of those in (coordinated) multiple views. Much of this has focused on fields related to visualization (data, scientific, information) and much of it is based around dataflow through a pipeline of processing stages and tools that allow specification of processing elements (source selectors, filters, mappers, renderers) and pipelines connecting them. The "animation production environment" (apE) [21] and AVS [5] permit scientific visualization workflows to be specified and realized using direct manipulation visual programming interfaces. ConMan [9] and VTK [22] provide a similar approach for graphics applications. In information visualization, the "data state model" [4], which represents workflow as a series of data transformations, has been influential and has been adopted by popular information visualization toolkits such as Prefuse [10].

Coordinated Multiple Views (CMVs) have the premise that users understand their data better if they interact with the information and view it through different perspectives [20]. They add the need for a coordination model [2] to the dataflow to permit users to interact in a coordinated way across a range of different visualizations. Snap [18], has a more data-centric approach to coordination, but provides a direct-manipulation visual language for building visualizations. Similar approaches include: GeoVISTA [23], GeoAnalytics [14], and VisTrails [1].

A number of meta-tools have been developed over many years to enable rapid development of graphical design environments [7, 8, 16]. Typically these are desktop tools themselves and are used to generate desktop IDE-hosted visual design tools rather than web-based visualization tools. However, the concept of such visual specification of visual language tools has proved useful in this domain.

Other types of application using dataflow and visual languages for manipulation are mashup generators like Yahoo Pipes[6] and (the now discontinued) Microsoft Popfly[7]. These applications allow various public feeds (e.g. RSS news feeds) and services (search engines, photo sharing sites, etc.) as data sources. Data manipulation uses operators such as filters or unions. The output consists e.g. of RSS feeds or mashups (for instance a map combined with geotagged images).

3. OUR APPROACH

To reduce the programming overhead of constructing Visual Wiki applications, we were motivated to develop a toolset through which we could straightforwardly specify and generate new Visual Wiki instances: the *VikiBuilder* meta-tool. We were attracted to the dataflow metaphor of much of the work described above as we felt this had a good closeness of mapping to the problem domain. We also felt the visual language based approaches, such as apE, AVS and Yahoo Pipes provided good productivity enhancement, and the base dataflow and co-ordination elements were appropriate for our domain.

Accordingly, we began by examining each of the Visual Wiki applications we had developed, abstracted from them a set of reusable generic processing elements, and defined "standard" APIs and parameters that could be used to integrate them together and customize them for specific purposes. Based on these generic elements, we derived a Domain Specific Visual Language (DSVL) for specifying combinations of elements with dataflow based workflow connections linking them together. An environment for this DSVL was then realized which supported the specification of Visual Wiki applications. From these specifications, code generators supported their realization reusing the generic elements and generated "glue code". This VikiBuilder Visual Wiki meta-tool is itself realized as a Visual Wiki application, providing its specification DSVL and textual information to users as a Visual Wiki.

4. VIKIBUILDER

4.1 A DSVL for Visual Wikis

From analysis of a range of our Visual Wikis, the generic types of processing element we identified are:

- Data Source: a data base, web service, flat file, etc.
- Adapter: describes how a data source is accessed. For example a web service could be accessed via its API. In that case the data access module describes this API.
- Data Representation: describes how the data is represented after access (or transformation).
- Transformation Agent: describes criteria e.g. for filtering a data representation or for merging data sources or data representations.
- View: describes the views within the frames.
- Coordination Object: sits between a view and another element; describes how events impact related elements.

Each of these generic processing element types correspond to a visual language element in the VikiBuilder DSVL. Instances of these elements are connected via dataflow connectors to specify a Visual Wiki application. Figure 3 shows these various visual language elements, together with VikiBuilder DSVL specifications of two of our Visual Wiki applications.

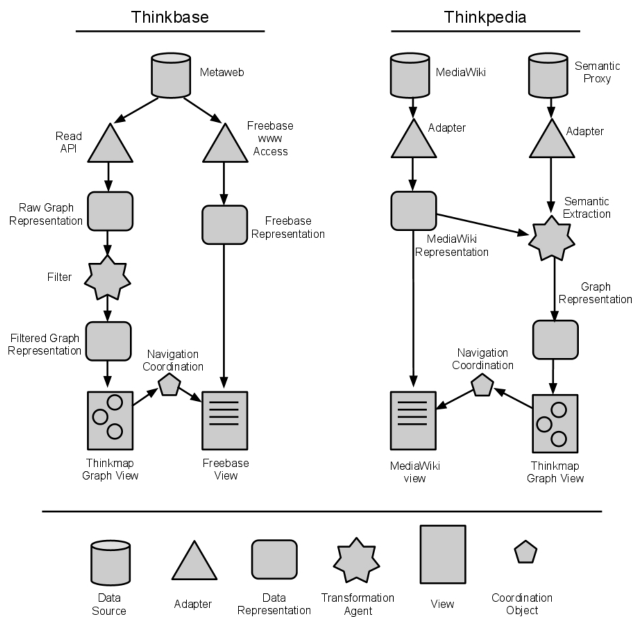


Figure 3. Thinkbase (left) and Thinkpedia (right) described using a dataflow metaphor.

[6] http://pipes.yahoo.com

[7] http://www.popfly.com

At left is the *Thinkbase* specification. This has one data source, the Freebase (Metaweb) data base. A data access adapter element uses an API to access the source (left side of flow) resulting in a "raw" graph representation of the source. This is filtered using a transformation agent (e.g. filtering out specific node types) and the result displayed in a Thinkmap graph view (i.e. data in graph format is passed to Thinkmap to create a visual representation). On the right side of the flow is the standard web access to the Metaweb data through the Freebase view. A navigation coordination element specifies that a navigation event in the Thinkmap view influences the Freebase view (e.g. handing over the ID of the new center node object).

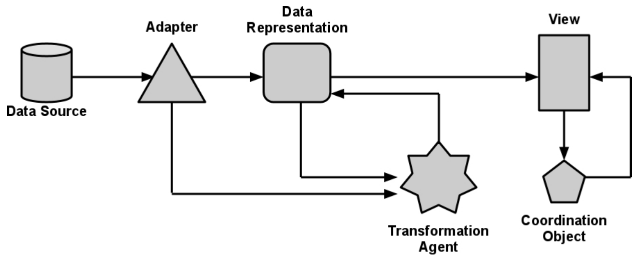


Figure 4. Overview of the VikiBuilder meta-model.

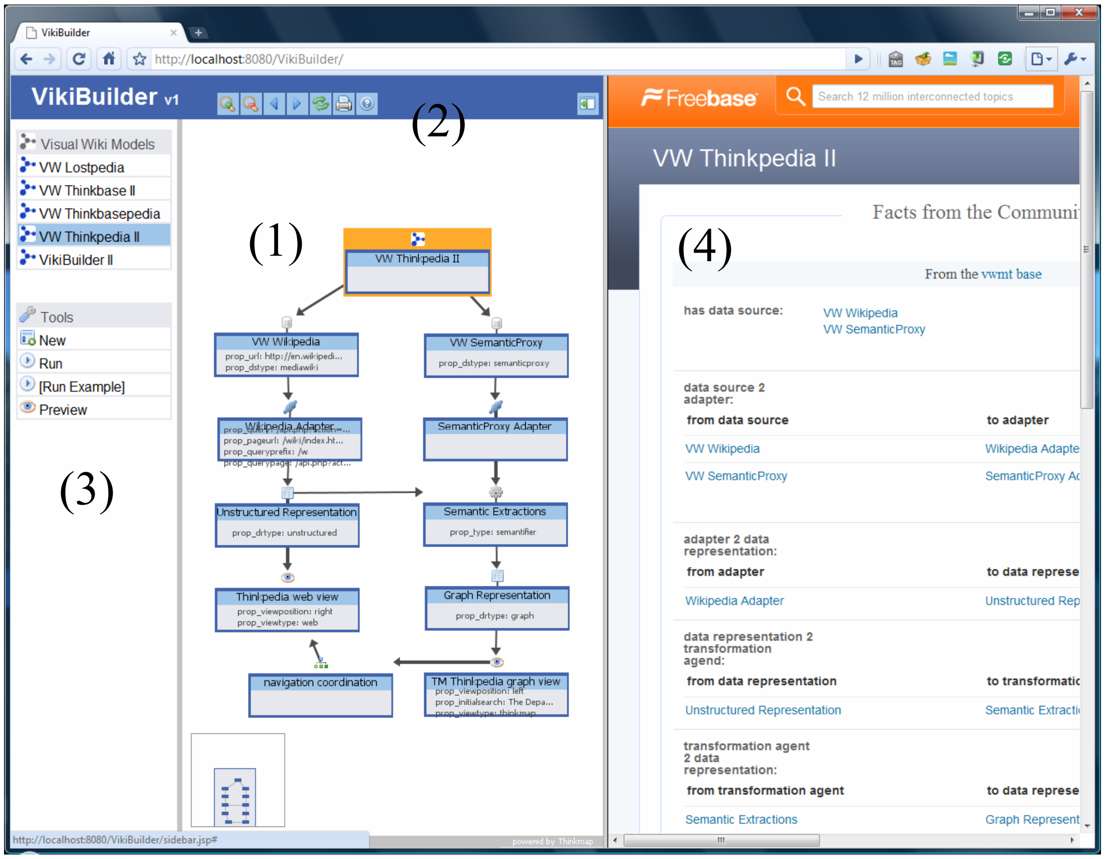
The *Thinkpedia* example in Figure 3 right uses two data sources, a MediaWiki and the SemanticProxy. A transformation agent specifies how the MediaWiki data is "semantified" using the SemanticProxy. The outcome is again a graph representation, which is displayed using a Thinkmap view. Additionally the MediaWiki web representation is displayed in a second view. Generalizing from these and other examples, Figure 4 is an overview of the meta-model for the VikiBuilder DSVL, expressed using the same DSVL notation. For clarity, many details, e.g. element parameters and properties, are omitted.

4.2 VikiBuilder: a Visual Wiki meta-tool

Figure 5 shows our VikiBuilder meta-tool in use. The tool provides an environment for designing Visual Wikis using our DSVL, together with code generation and preview facilities that allow users to generate Visual Wikis from the specification and preview them as changes to the specification are made.

In Figure 5, VikiBuilder is being used to specify a Thinkpedia style tool using a visual specification similar to Figure 3 (right). A DSVL editor (1) allows modeling and visualization of the Visual Wiki design, with a tool bar (2) and project management facilities (3) at left. Details of the visual elements (specific element values, parameters, etc.) used in the DSVL model are specified in a Freebase view (4). The DSVL and Freebase views are coordinated, meaning the application is itself a Visual Wiki. As one can see we have had to make some compromises in the appearance of the DSVL (1) due to limitations in the Thinkmap visualization engine, e.g. use of icons instead of shapes (compare with Figure 3).

The modeling application uses Freebase as data storage and Thinkmap for the visual interface. The tool allows the creation, storage and editing of Visual Wiki architectures which are



VW Thinkpedia II

Facts from the Communi

From the vwmt base

has data source: VW Wikipedia, VW SemanticProxy

data source 2 adapter:

from data source	to adapter
VW Wikipedia	Wikipedia Adapte
VW SemanticProxy	SemanticProxy Ad

adapter 2 data representation:

from adapter	to data represe
Wikipedia Adapter	Unstructured Rep

data representation 2 transformation agend:

from data representation	to transformatic
Unstructured Representation	Semantic Extracti

transformation agent 2 data representation:

from transformation agent	to data represe
Semantic Extractions	Graph Represent

Figure 5. VikiBuilder meta-tool in use.

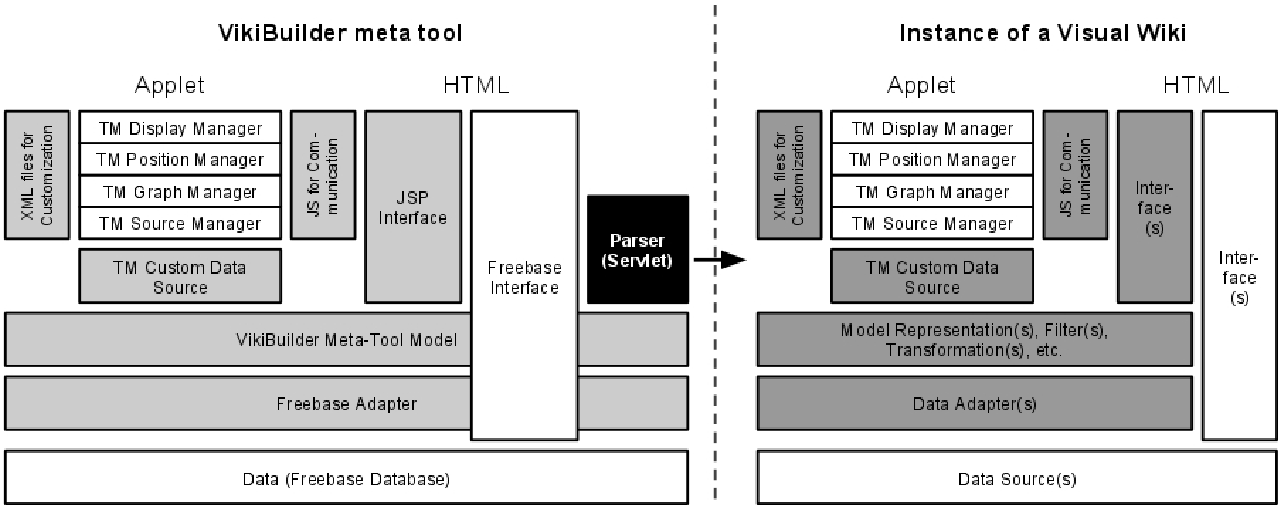


Figure 6. The production of a Visual Wiki instance (right) based on a model developed in the meta-tool (left).

stored in a Freebase domain. The schema of the Freebase domain represents the meta-model of the DSVL, elaborating the model shown in Figure 4. In designing the application our aim was to (1) make the application as easy to use as possible and (2) to use a metaphor for construction that would appeal to Visual Wiki designers. Constructing the application as a Visual Wiki itself supported both of these design aims.

Once a Visual Wiki application has been specified, the meta-tool can automatically generate the Visual Wiki instance. This takes the visual specification and compiles it to appropriate code and wiki templates to construct the application. The VikiBuilder tool also provides Preview support for generated Visual Wikis. In Figure 7 we can see the preview facilities in use. Here a Visual Wiki, similar to Thinkpedia, is being developed for the Lostpedia[8] wiki, a wiki about the *Lost* TV show. The visual specification for the Visual Wiki is shown at left (1). The other design views (Freebase and toolbar) have been elided to provide screen space for the preview. The "VW Lostpedia" Visual Wiki preview is shown in the frames at the right. This Visual Wiki has a Thinkmap visualization at the top (2) and a Lostpedia wiki view at the bottom (3). The preview can be used to test out the design decisions made in the Visual Wiki design prior to deployment of the new Visual Wiki application.

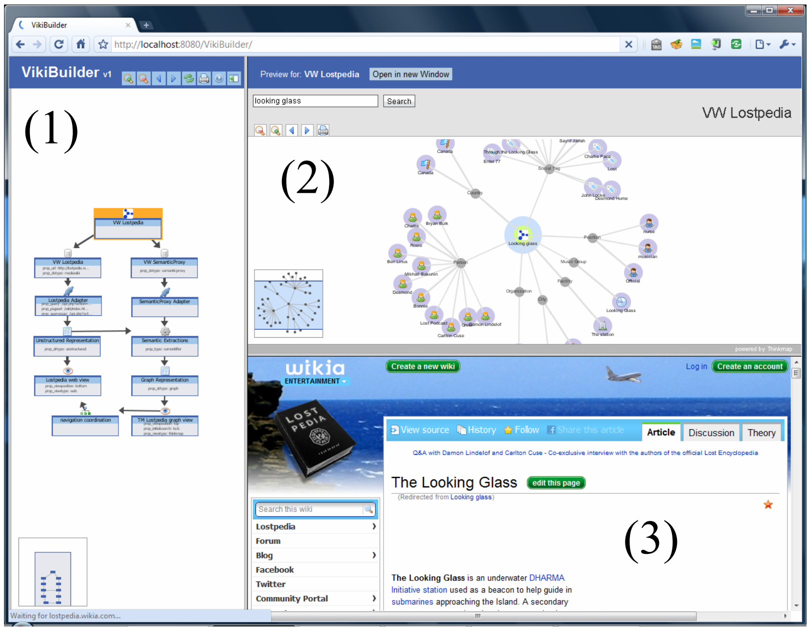
4.3 Architecture and implementation

The functionality of the meta-tool can roughly be divided into two main areas or steps:

1. Functionality to allow a user to model a Visual Wiki. This includes an interface to create, change, and save models.
2. Functionality to transform this model into an instance of a Visual Wiki: process models (from 1) to automatically create code, property files, etc, and compile and present them.

Step 1 is similar to previous implementations (e.g. Thinkbase) and will be described first. This will also include a brief description of the architecture as well as the directory system, as those will be extended in step 2.

Figure 6 left shows the different layers of the pipeline architecture of the meta-tool (for step 1). From bottom to top: Freebase is the data source for the application. A customized "domain" in Freebase supports modeling Visual Wikis. The Freebase adapter makes use of the Freebase API and provides access to the data in a convenient way. The Visual Wiki meta-tool model is an internal representation of the model retrieved from the Freebase database. The model is turned into an interactive visual representation using Thinkmap (left side). This includes a custom data source layer, which translates the Model into the format required by Thinkmap. The remaining



The Looking Glass is an underwater DHARMA Initiative station used as a beacon to help guide in submarines approaching the Island. A secondary

Figure 7. VikiBuilder preview for Lostpedia.

[8] http://lostpedia.wikia.com

Thinkmap layers (Source, Graph, Position, and Display) are provided by the Thinkmap SDK. However, they are customized via XML files (e.g. their behavior and appearance). To the right, a JSP interface accesses the Visual Wiki meta-tool Model to create parts of the application's user interface. The default Freebase interface is currently used to edit the model.

Step 2 takes the model of a Visual Wiki (stored in Freebase) and turns it into an actual instance of a Visual Wiki. This is implemented using Servlets, which parse the model, create code and compile the new instance. A Parser Servlet (Figure 6, centre) is used to generate the new Visual Wiki instance (Figure 6, right) as follows:

1. The model is traversed and all the needed directories and files are created based on the properties of the model entities. These include: User interface specifications, including JSP and XML property files; back-end logic code such as data source access and data transformation; and an XML build specification file.
2. After all the necessary files have been created, an automatically invoked build tool uses the XML build file to construct the new Visual Wiki instance (compile and deploy the code, etc).

This process generates all of the darker components shown in Figure 6 (right) and links them to the other preexisting components, such as the data sources.

5. EXAMPLES

5.1 Thinkpedia II

In order to test the VikiBuilder meta-tool and prove the concept of our tool, we have modeled and created several Visual Wiki instances. As a first example, we have re-built our Thinkpedia prototype (see Motivation section) in the form of *Thinkpedia II*, i.e. a Visual Wiki, which has the same basic design features as one of our originally hand-crafted applications. As described in Section 2, Thinkpedia is a visual exploration tool for Wikipedia. It uses the SemanticProxy service to extract semantically enriched concepts out of Wikipedia articles, visualizes those in an interactive graph, and therefore provides a visually appealing way to browse and explore the vast Wikipedia contents. To realize a re-implementation of Thinkpedia, we started by describing the model of the application in our VikiBuilder visual language. The final model (described theoretically in Figure 3) is shown in Figure 8.

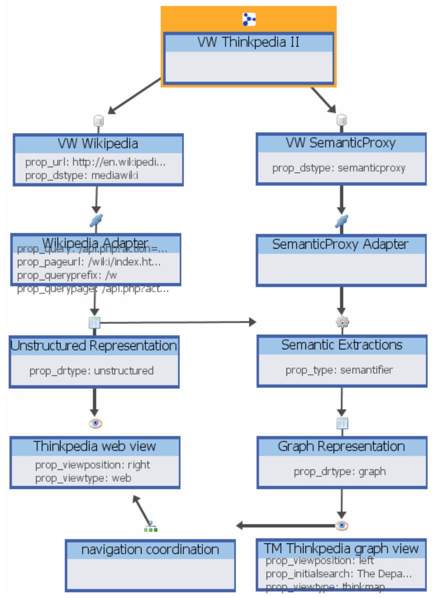


Figure 8. The Thinkpedia II model created in VikiBuilder.

The key entities of the model are: Wikipedia and SemanticProxy as the data sources; adapters, which access them (through their respective APIs); a transformation agent, which combines the "raw" content from Wikipedia with the

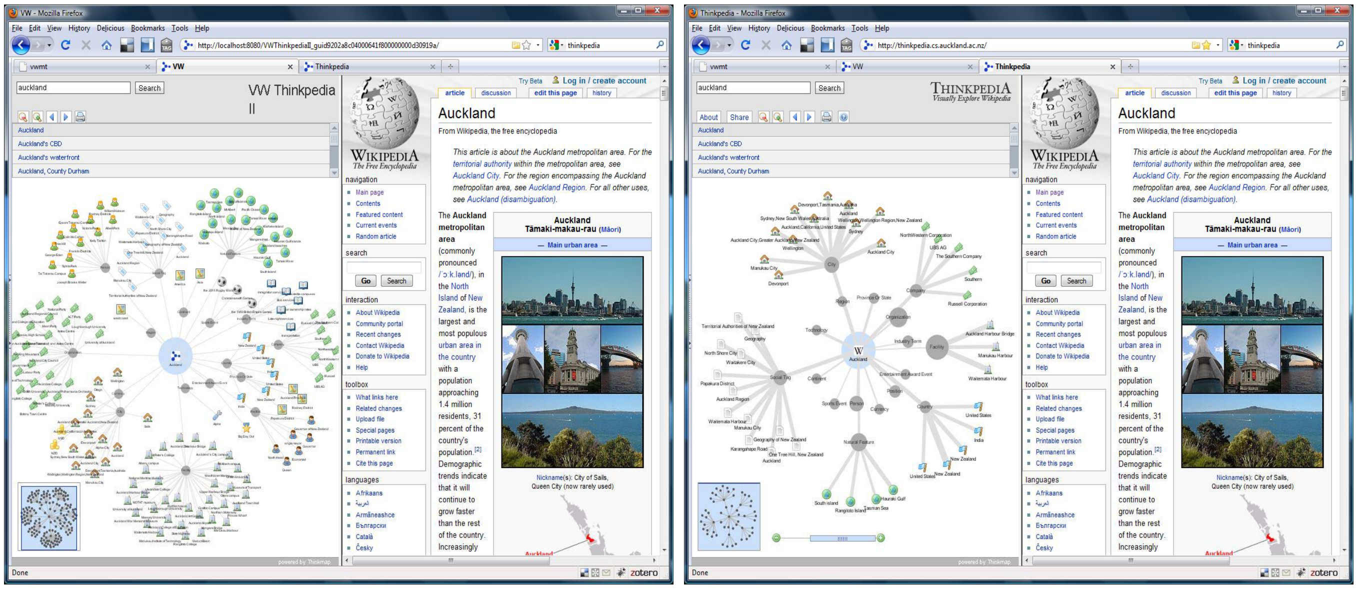

VW - Mozilla Firefox
VW Thinkpedia II
Thinkpedia - Mozilla Firefox
Thinkpedia
Visually Explore Wikipedia
Auckland
From Wikipedia, the free encyclopedia
WIKIPEDIA
The Free Encyclopedia

Figure 9. Thinkpedia II (left) and the original hand-crafted Thinkpedia (right).

SemanticProxy; a resulting structured data representation; a Thinkmap view, which further processes and visualizes this data representation; a standard Wikipedia view; and a coordination object, which defines the navigation behavior. All of these entities have properties which describe the specifics of Thinkpedia II. For instance the Wikipedia data source and the adjacent adapter entities describe how the application makes use of Wikipedia. Properties include e.g. the URL of the wiki and the MediaWiki API (e.g. search API and its parameters). Properties of the view entities include, amongst others, definitions of the type of view (e.g. Thinkmap) and the positioning of the frames. The creation of this model can be done without any need for programming through the form-based editing interface of Freebase (see Figure 5).

After the model has been created, we can hit the "Run" button, and Thinkpedia II is automatically created as a new Visual Wiki instance. As described in Section 4.3, this is done by parsing the model which will automatically create code and property files based on the entities and their parameters in the model. Finally, the code is compiled. The outcome is a new stand-alone instance of a Visual Wiki, which is accessible through its own URL. The application specification can be loaded into the VikiBuilder again and further refined.

Figure 9 shows the final Thinkpedia II Visual Wiki adjacent to our original hand-crafted Thinkpedia. The new Thinkpedia II can be used in just the same way as the original application. An interactive graph of a selected Wikipedia article is visualized next to that article. A user can explore and navigate the wiki space via the graph, by clicking on nodes, which will update the visualization as well as the wiki view. All the basic features, such as navigation, search, and browsing history are the same as in the original Thinkpedia. Some of the more advanced features of our customized Visual Wikis are not possible to specify and implement in the current version of the VikiBuilder. In the case of Thinkpedia II, these include the visualization of semantic relevance (reflected in edge thickness), advanced filtering mechanisms, and a sharing feature for the graph view.

5.2 Lostpedia Visual Wiki

After showing the feasibility of re-implementing one of our original prototypes, we explored modeling and automatically creating different variations of Thinkpedia II style Visual Wikis. One of these is *VW Lostpedia*, which can be seen in Figure 10.

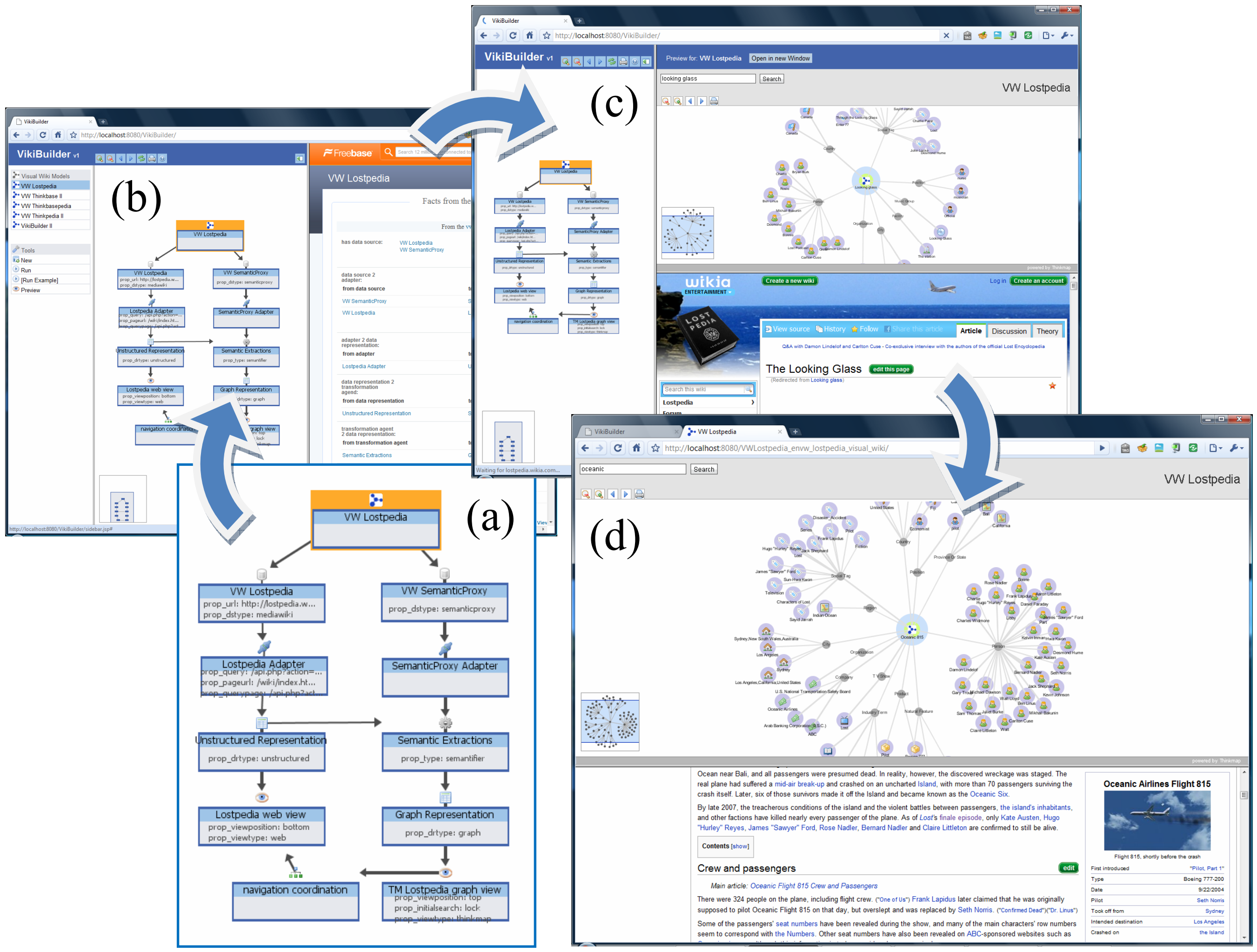
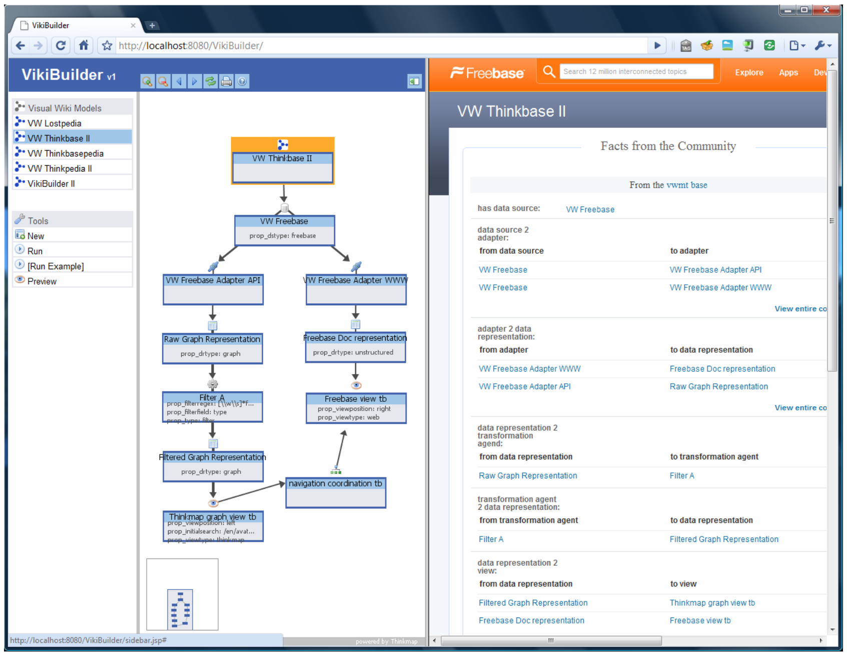


Figure 10. Lostpedia: The complete building process from (a) the model, to (b) the specification of the model, to (c) a preview of the Lostpedia Visual Wiki, to (d) the final stand-alone instance of the new Visual

VW Lostpedia is a visual exploration tool for Lostpedia, a wiki about the *Lost* TV show. We started modeling this Visual Wiki instance by taking the model from Thinkpedia II as a starting point: models can be copied and renamed to support simple reuse. Figure 10 (a) shows the final model for VW Lostpedia. It is similar to Thinkpedia II as it uses a MediaWiki-based wiki as a data source, extracts semantic meaning out of it utilizing the SemanticProxy, and visualizes information with Thinkmap. The main difference is that we are using a different wiki. As a user this can be realized by specifying different URLs and API parameters for the data source and adapter entities. Furthermore, we changed several parameters of the view entities so that they would be displayed in different frame locations. The complete production life-cycle of the VW Lostpedia can be seen in Figure 10: After specifying (a) and editing the model in the Freebase view (b), we can preview the application (c), and finally produce the stand-alone instance of a new Visual Wiki (d). By modeling and implementing VW Lostpedia, we have shown that we can also produce a new Visual Wiki rather than just replicating an existing one. Currently we are able to rapidly model and automatically create similar customized Visual Wiki variations with virtually any MediaWiki-based wiki in typically less than a day.

5.3 Thinkbase II

As a third example we re-modeled another one of our prior Visual Wiki implementations, Thinkbase (see Figure 2), which allows visual exploration of the semantic wiki Freebase by extracting topics and their semantic relationships using the Freebase API and visualizing them in an interactive graph representations. Our goal was to model and automatically create a Visual Wiki with the same basic functionality: *Thinkbase II*. We have created the necessary dataflow model for Thinkbase II inside the VikiBuilder, see Figure 11. The application has only one data source, the Freebase Metaweb database, which on the one hand is accessed through the standard Freebase user interface, and on the other side through its API. The raw graph which is retrieved from that API is first filtered (certain entities will be filtered out) and then handed over to the Thinkmap framework to create the visualization. Detailed parameter specifications in the case of Thinkbase II include the way the Freebase API is accessed, how the data format of the source is converted into an internal representation, and on what criteria entities are filtered out. Figure 12 shows the final Thinkbase II Visual Wiki instance. The application has the same basic features as the original hand-crafter version of Thinkbase. The Freebase contents can be explored visually by navigating along the graph. The filter element even provides a new and more rapid way of altering the appearance of the graph. For instance, Figure 12



VW Thinkbase II

Facts from the Community

From the vwmt base

has data source: VW Freebase

data source 2 adapter:

from data source	to adapter
VW Freebase	VW Freebase Adapter API
VW Freebase	VW Freebase Adapter WWW

View entire co

adapter 2 data representation:

from adapter	to data representation
VW Freebase Adapter WWW	Freebase Doc representation
VW Freebase Adapter API	Raw Graph Representation

View entire co

data representation 2 transformation agend:

from data representation	to transformation agent
Raw Graph Representation	Filter A

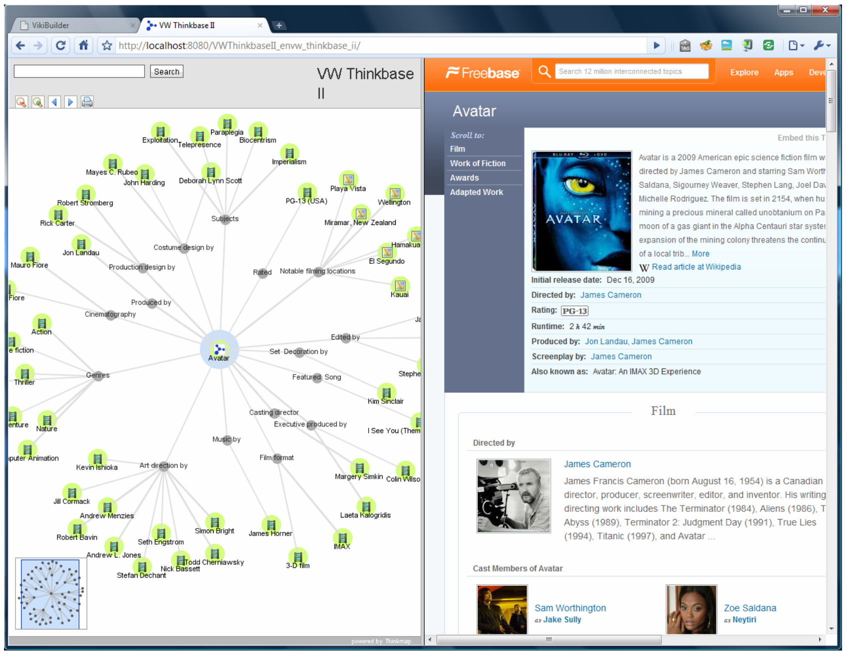
transformation agent 2 data representation:

from transformation agent	to data representation
Filter A	Filtered Graph Representation

data representation 2 view:

from data representation	to view
Filtered Graph Representation	Thinkmap graph view tb
Freebase Doc representation	Freebase view tb

Figure 11. The Thinkbase II model in VikiBuilder.



Avatar

Avatar is a 2009 American epic science fiction film w directed by James Cameron and starring Sam Worth Saldana, Sigourney Weaver, Stephen Lang, Joel Da Michelle Rodriguez. The film is set in 2154, when hu mining a precious mineral called unobtanium on Pa moon of a gas giant in the Alpha Centauri star syste expansion of the mining colony threatens the continu of a local trib... More

W Read article at Wikipedia

Initial release date: Dec 16, 2009
Directed by: James Cameron
Rating: PG-13
Runtime: 2 h 42 min
Produced by: Jon Landau, James Cameron
Screenplay by: James Cameron
Also known as: Avatar: An IMAX 3D Experience

Film

Directed by

James Cameron

James Francis Cameron (born August 16, 1954) is a Canadian director, producer, screenwriter, editor, and inventor. His writing directing work includes The Terminator (1984), Aliens (1986), T Abyss (1989), Terminator 2: Judgment Day (1991), True Lies (1994), Titanic (1997), and Avatar ...

Cast Members of Avatar

Sam Worthington *as* Jake Sully

Zoe Saldana *as* Neytiri

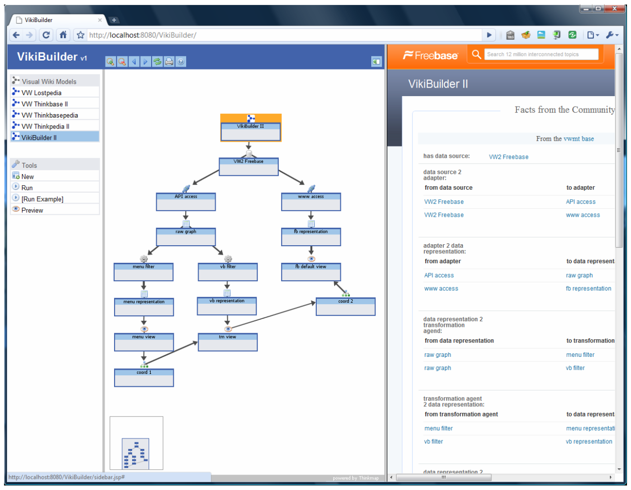
Figure 12. Thinkbase II Visual Wiki.

shows the application where all entities which are not related to the type "Movie" are filtered out. However, there are also still some limitations compared to the original version. For instance, some specific types of nodes (e.g. URL nodes) are currently not displayed.

Our experience with these three exemplar Visual Wiki instances, which we are able to model and generate, has, we believe, successfully proven the concept of the VikiBuilder. We were able to rapidly re-create previously hand-crafted Visual Wiki instances including all of their core functionalities, as well as new variations of them. Both our visual language to model Visual Wikis as well as the code generator are flexible and readily extendible, so that they can be easily refined in our future work to allow a broader range of Visual Wikis to be specified and generated.

5.4 VikiBuilder II

As a final example we attempted to model our own VikiBuilder (which is itself a Visual Wiki) inside the VikiBuilder meta-tool. The outcome of this *VikiBuilder II* model can be seen in Figure 13. As with our Thinkbase tool, the VikiBuilder II model specifies Freebase as its data source. On the one side, this data source is access via the API, split up into two flows, filtered using different criteria, and finally displayed in two different views, a menu view (for navigating and managing the different Visual Wiki models) and a Thinkmap graph view. On the other side, the Freebase content is shown using its default web interface. The three views are coordinated. In case of VikiBuilder II, the model is not yet able to automatically produce a finished Visual Wiki instance, as the VikiBuilder is much more complex compared to our other applications. This is especially the case for the code generation functionality, which we are not yet able to model in our VikiBuilder visual language. Extending our modeling language as well as the code generator in order to support these and similar advanced features will be part of our future work.



has data source: VW2 Freebase

data source 2 adapter:

from data source	to adapter
VW2 Freebase	API access
VW2 Freebase	www access

adapter 2 data representation:

from adapter	to data represent
API access	raw graph
www access	fb representation

data representation 2 transformation agend:

from data representation	to transformation
raw graph	menu filter
raw graph	vb filter

transformation agent 2 data representation:

from transformation agent	to data represent
menu filter	menu representati
vb filter	vb representation

data representation 2

Figure 13. VikiBuilder II modeled inside VikiBuilder.

6. DISCUSSION

Our prior work developing hand-crafted Visual Wikis and their evaluations have successfully demonstrated the usefulness and appeal of the Visual Wiki concept [12, 13]. VikiBuilder provides the ability to develop Visual Wiki applications rapidly with minimal hand-crafting. This has been demonstrated by the success we have had in replicating implementations of our earlier Visual Wikis, together with additional new Visual Wiki applications, such as the Lostpedia Visual Wiki, using VikiBuilder. In each case, the time taken for implementation was of the order of days (or shorter), rather than the weeks or months of the prior applications. Rather than having to focus on technical programming issues, we were able to focus on more creative issues, such as placement of frames and so on, with rapid feedback on changes to the design possible via the preview facilities.

We feel our decision to design VikiBuilder as a Visual Wiki itself was a valid one. This approach provides good "closeness of mapping" [6] to the pipeline metaphor that most information visualization designers use. In addition, as with most maturing frameworks, having identified a library of reusable computational artifacts, the bulk of the effort in constructing a Visual Wiki application is in configuring parameters for these components, instantiating them, and writing "glue code" to connect them together [19]. The choice of Freebase for underlying data representation and storage suited these needs well. The frame based representational model of this semantic wiki lends itself naturally to describing templates for parameterized artifacts, and instantiations of them in a readable, readily editable format as Freebase pages. The usual disadvantage of such an approach, the hidden dependencies [6] created by such a plethora of small artifact descriptions, is obviated via the context view of the visual language specification, which provides both an overview, and the primary mechanism for specifying and understanding the "glue code" connections between artifacts. This is an example of Moody's Principle of Complexity Management, which advocates such a hierarchical decomposition of notational views [17]. The coordination mechanism between the visual and Freebase views supports Moody's Principle of Cognitive Integration; providing explicit linkages between the different notational diagrams [17].

Weaknesses of our current VikiBuilder implementation include the following. Firstly, and most importantly, the range of Visual Wiki elements is currently limited. We would like to expand VikiBuilder to permit Visual Wiki implementations using other components, e.g. other visualization interfaces, such as Prefuse[9], other semantifier services, various analysis services and additional data sources, such as adaptors for other wiki and database APIs. These are all straightforward programming tasks and will require minimal change to the existing tool architecture, and some minimal change to the code generation facilities.

Secondly, we have had to make some compromises in implementing the DSVL due to limitations in the Thinkmap visualization engine. For example, Moody argues that shape is one of the most important distinguishing characteristics for visual notational element design. We adopted this in the draft DSVL design, as seen in Figure 3, but were forced to use other visual channels, such as iconic annotations and color, in the Thinkmap realization. Other minor usability issues also result from such compromises. Expansion of the range of components noted above opens up the possibility of re-implementing VikiBuilder using itself, to effectively port VikiBuilder to other front-end technologies obviating some of these issues.

In future work, we plan to extend the range of components available to implement Visual Wikis. This includes incorporation of new instances of existing component types, as outlined above, but also extensions to the VikiBuilder DSVL to accommodate new types of element, such as a more refined ability to express and realize semantic analysis workflows, and a

[9] http://prefuse.org/

broader range of annotations, such as the semantic strength link widths in the original Thinkpedia. These latter extensions are obviously more complex and will require extensions to the VikiBuilder meta-model, editing tools and code generation facilities. We also plan to ameliorate various minor usability issues we have identified in the existing implementation.

We are obviously keen to use VikiBuilder to construct new Visual Wiki applications. The success of our earlier Visual Wiki exemplars has led to strong commercial interest in our approach. To satisfy this interest, we need to be in a position to move from craft to production: VikiBuilder provides us the mechanism to do so. In addition we are keen to further explore the broader potential of the Visual Wiki concept and we see VikiBuilder as a platform for us to more rapidly realize this, in the same way that our work on meta-tools for Visual Language design and implementation [7, 8] have assisted us in that latter domain.

7. CONCLUSIONS

We have introduced VikiBuilder, a Visual Wiki application which permits rapid specification and realization of Visual Wiki tools. The latter combine visualizations, to provide a context overview and contextual navigation, with complex wikis, where individual wiki pages provide more focused detail on topics of interest. Our prior work has demonstrated the utility and appeal of such applications. VikiBuilder provides a meta-tool to more rapidly design and implement them. We have demonstrated the utility of VikiBuilder by recreating several of our prior Visual Wiki applications and implementing new ones. This showed a very significant productivity gain over hand-crafted solutions. By extending the range and type of components able to be instantiated we will extend the range of Visual Wiki applications we can support.

ACKOWLEDGEMENTS

The authors acknowledge the assistance of the BuildIT Doctoral Scholarship fund and the FRST Software Process and Product Improvement project.

REFERENCES

[1] Bavoil, L., et al., *VisTrails: Enabling Interactive Multiple-View Visualization*. In Proc. of IEEE Visualization, 2005.

[2] Boukhelifa, N. and P.J. Rodgers, *A model and software system for coordinated and multiple views in exploratory visualization*. Information Visualization, 2: 258-269, 2003.

[3] Buffa, M. and F. Gandon, *SweetWiki: Semantic Web Enabled Technologies in Wiki*. Proceedings of International Symposium on Wikis, 2006.

[4] Chi, E.H., *A Taxonomy of Visualization Techniques using the Data State Reference Model*. Proc. of InfoVis, 2000.

[5] Dyer, D.S., *A Dataflow Toolkit for Visualization*. IEEE Computer Graphics and Applications. 10: 60-69, 1990.

[6] Green, T.R.G. and M. Petre, *Usability Analysis of Visual Programming Environments: A 'Cognitive Dimensions' Framework*. Journal of Visual Languages and Computing, 7:131-174, 1996.

[7] Grundy, J.C., Hosking, J.G., Li, N. and Huh, J. *Marama: an Eclipse meta-toolset for generating multi-view environments*, Formal demonstration at the 30th International Conference on Software Engineering (ICSE 2008), Leipzig, Germany, May 2008, ACM Press.

[8] Grundy, J.C., Hosking, J.G., Zhu, N. and Liu, N., *Generating Domain-Specific Visual Language Editors from High-level Tool Specifications*. Proceedings of the 2006 IEEE/ACM International Conference on Automated Software Engineering, Tokyo, 24-28 Sept 2006, IEEE.

[9] Haeberli, P.E., *ConMan: a visual programming language for interactive graphics*. ACM SigGraph Computer Graphics, 22:103-111, 1988.

[10] Heer, J., S.K. Card, and J.A. Landay, *Prefuse: a toolkit for interactive information visualization*. ACM, 2005.

[11] Hirsch, C., Grundy, J.C., and Hosking, J.G., *Thinkbase: A Visual Semantic Wiki*. Demo Session of the 7th International Semantic Web conference, 2008.

[12] Hirsch, C., Hosking, J.G., and Grundy, J.C., *Interactive Visualization Tools for Exploring the Semantic Graph of Large Knowledge Spaces*. 1st Int'l Workshop on Visual Interfaces to the Social and the Semantic Web, 2009.

[13] Hirsch, C., Hosking, J.G., Grundy, J.C., Chaffe, T., MacDonald, D., and Halytskyy, Y, *The Visual Wiki: A New Metaphor for Knowledge Access and Management*. Proc. of the 42nd Hawaii International Conference on System Sciences, 2009, IEEE CS Press.

[14] Johansson, S. and M. Jern., *GeoAnalytics visual inquiry and filtering tools in parallel coordinates plots*. Proc. of the 15th Annual ACM International Symposium on Advances in Geographic Information Systems, 2007.

[15] Majchrzak, A., C. Wagner, and D. Yates., *Corporate wiki users: results of a survey*. Proceedings of International Symposium on Wikis, 2006.

[16] Mazanek, S., S. Maier, and M. Minas, *Auto-completion for Diagram Editors based on Graph Grammars.* Proc. of the 2008 IEEE Symposium on Visual Languages and Human-Centric Computing. IEEE CS Press, 2008.

[17] Moody, D.L., *The 'Physics' of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering*. IEEE Transactions on Software Engineering, 2009.

[18] North, C., et al., *Visualization schemas and a web-based architecture for custom multiple-view visualization of multiple-table databases*. Information Visualization, 2002.

[19] Roberts, D. and R. Johnson, *Evolving Frameworks: A Pattern Language for Developing Object-Oriented Frameworks*. Proc. of PLoP, 1996.

[20] Roberts, J.C., *State of the art: Coordinated & multiple views in exploratory visualization*. ETH, Switzerland, IEEE Press, 2007.

[21] Schröder, F., *apE—the original dataflow visualization environment*. ACM SigGraph Computer Graphics, 1995.

[22] Schroeder, W.J., K.M. Martin, and W.E. Lorensen, *The design and implementation of an object-oriented toolkit for 3D graphics and visualization*. IEEE Visualizations 1996.

[23] Takatsuka, M. and M. Gahegan, *GeoVISTA Studio: A codeless visual programming environment for geoscientific data analysis and visualization*. Computers and Geosciences, 28:1131-1144, 2002.

[24] Ting, M.S., Hirsch, C., and Hosking, J.G., *KaitoroBase: Visual Exploration of Software Architecture Documents*, Formal demonstration at ASE, 2009.