Velo: Riding the Knowledge Management Wave for Simulation and Modeling

Ian Gorton, Chandrika Sivaramakrishnan, Gary Black, Signe White, Sumit Purohit, Michael Madison, Karen Schuchardt
Pacific Northwest National Laboratory
Richland, WA, USA
509-375-3850
ian.gorton@pnl.gov

ABSTRACT
Modern scientific enterprises are inherently knowledge-intensive. In general, scientific studies in domains such as geosciences, climate, and biology require the acquisition and manipulation of large amounts of experimental and field data in order to create inputs for large-scale computational simulations. The results of these simulations must then be analyzed, leading to refinements of inputs and models and additional simulations. Further, these results must be managed and archived to provide justifications for regulatory decisions and publications that are based on these models. In this paper we introduce our Velo framework that is designed as a reusable, domain independent knowledge management infrastructure for modeling and simulation. Velo leverages, integrates, and extends open source collaborative and content management technologies to create a scalable and flexible core platform that can be tailored to specific scientific domains. We describe the architecture of Velo for managing and associating the various types of data that are used and created in modeling and simulation projects, as well as the framework for integrating domain-specific tools. To demonstrate a realization of Velo, we describe the Geologic Sequestration Software Suite (GS^3) that has been developed to support geologic sequestration modeling. This provides a concrete example of the inherent extensibility and utility of our approach.

Categories and Subject Descriptors
H.3.5 [**INFORMATION STORAGE AND RETRIEVAL**]: Online Information Services - *Data sharing, Web-based services.*

General Terms
Design, Management, Documentation

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SE-CSE'11, May 28, 2011, Waikiki, Honolulu, HI, USA
Copyright 2011 ACM 978-1-4503-0598-3/11/05 ...$10.00

Keywords
Knowledge Management, content management, scientific modeling, simulation, wiki

1. INTRODUCTION
Simulation, along with theoretical and empirical studies, is firmly entrenched in many areas of science [1]. As the scale of computational models and simulations grow to address ever more complex challenges [2], it is becoming increasingly difficult to create simulation inputs and manage and analyze the results [3]. Several distinct scientific disciplines such as climate [4] and geosciences [5] have made significant investments in creating successful comprehensive data management and analysis systems. These however, have limited applicability to other scientific domains, or do not support the full modeling and simulation lifecycle.

Much effort has been targeted at creating various generic technologies to support modeling and simulation across many science domains. For example, a number of approaches to support data provenance [6], workflows [7] and scalable distributed data management exist [8]. Many of the resulting technologies are deployed to provide point solutions for various user communities. Still, we are unaware of successful efforts to build a broadly applicable platform that can be easily tailored and used by different modeling and simulation domains.

In this paper we describe our work to design and build a flexible foundational technology known as *Velo* that can be used in modeling and simulation projects to:

- Capture, organize, query, and share experimental and observational data along with the tacit knowledge that is used to develop computational models
- Provide versioning of model inputs for specific projects and association of outputs with specific input data set versions
- Enable simulations to be launched on remote computational platforms (including HPC)
- Support both tight and loose integration of 3rd party tools to facilitate various modeling activities such as model development and visualization

- Provide a collaborative user interface that can be easily tailored by users to meet the needs of specific scientific domains

We refer to Velo as a knowledge management platform [9] for modeling and simulation. A knowledge management system (KMS) is an information and communication technology system developed to support and enhance explicit and tacit organizational knowledge [10]. For leading-edge organizations, KMSs are becoming a key factor in their success [11], thus contributing to wider deployment over a broad range of domains [12].

The Velo knowledge management platform is designed around a novel integration of a collaborative Web-based environment and a scalable enterprise content management system. In this paper we describe the Velo architecture and its realization, and show how the user environment can be customized for use in modeling and simulation studies that support geologic sequestration of greenhouse gases.

2. RELATED WORK

Contemporary Laboratory Information Management Systems (LIMS) such as [23][24][25] share some common aims and traits with Velo. LIMS must manage large data sets sourced from experimental facilities, and make the resulting data sets accessible to their user communities. However, LIMS typically provide little support for modeling and simulation activities that leverage the data sets they manage, and focus on automating well-defined workflows for processing experimental data. In contrast, Velo is defined to support the more ad hoc, incremental and iterative user-driven activities that are common in modeling and simulation projects.

Several data management systems for scientific data exist, for example, the Storage Request Broker (SRB) [26]. The SRB supports controlled sharing, publication, replication, transfer, attribute based organization, discovery, and preservation of distributed data. These technologies are complementary in that they can be used to provide the large-scale distributed data management capabilities that Velo requires. The files that are stored can then be transparently linked to and accessed through the Velo user environment.

At PNNL we have considerable experience in building first-of-a-kind scientific user environments for computational chemistry [13] and bioinformatics [15]. These widely used technologies are targeted specifically at the respective domain scientists, and exploit a variety of off-the-shelf (e.g. Postgres, J2EE, WebDAV) and custom built components. We also have experience with leveraging semantic technologies for knowledge management in science with the Scientific Annotation Middleware technology [14]. Our efforts in Velo differ from these in that we are attempting to build a more generic scientific knowledge management architecture that can be easily tailored to specific scientific domains.

Standard semantic wiki-based environments are useful for describing or documenting models and experiments [18] [19] but do not integrate with tools that assist with model execution or comparison. Model execution and comparison tools typically operate at a low level and do not facilitate collaboration and knowledge sharing. Velo aims to couple the advantages of semantic wiki-based environment and modeling tools by integrating them into a collaborative framework.

Other efforts are using wiki technologies as the core architecture for building new knowledge management services. For example, [16] describes three prototypes for enhancing a wiki's textual content with a complementary visual representation of the knowledge, and [17] uses wikis as the basis for passive visualization of social networks graphs.

Several knowledge base architectures have been proposed, for example [20][21], and a classification of these is given in [22]. Our approach shares some concepts and technologies with existing systems, but is differentiated by its attempt to leverage off-the-shelf software whenever possible. This provides a more timely, agile and robust solution.

3. Velo ARCHITECTURE

A large range of functional requirements have driven the design of Velo. These were derived from working with scientists in different modeling and simulation domains, for example carbon sequestration [29], and generalizing the findings to capture the common needs. From these discussions, some of the key drivers for the Velo architecture that emerged were as follows:

- Support distributed, collaborative teams of modelers working on common problems
- Support integration of external, off-the-shelf tools
- Capture field and experimental data associated with a modeling project
- Manage the evolution of models and their associated results over time
- Exploit mature open source technologies to ensure a robust, scalable platform

An overview of the resulting Velo architecture is depictured in Figure 1. The design provides a clean separation of concerns between managing the user environment and its associated data, and storing and managing the content associated with models and simulations. This loose coupling ensures that the user environment and content management systems can evolve independently to exploit advances in their underlying technologies. It also allows customization of the user environment for specific modeling and simulation disciplines without impacting the underlying management of the associated modeling data.

This core architecture provides other generic capabilities for incorporating third party tools into the user environment, and ingesting external data into Velo. These mechanisms are extensible so that discipline-specific tools and data can be seamlessly integrated and packaged for scientists to use.

The following sections explain in detail the design and implementation of Velo.

3.1 Velo Core

The Velo collaborative user environment is built upon MediaWiki[1]. MediaWiki is a widely deployed Web-based technology, with a stable open source codebase, extensible architecture, and a wealth of available extensions that, for example, provide enhanced role-based security. It provides a PHP

[1] http://www.mediawiki.org/wiki/MediaWiki

scripting language based environment for customization, and stores wiki contents in a relational database (MySQL).

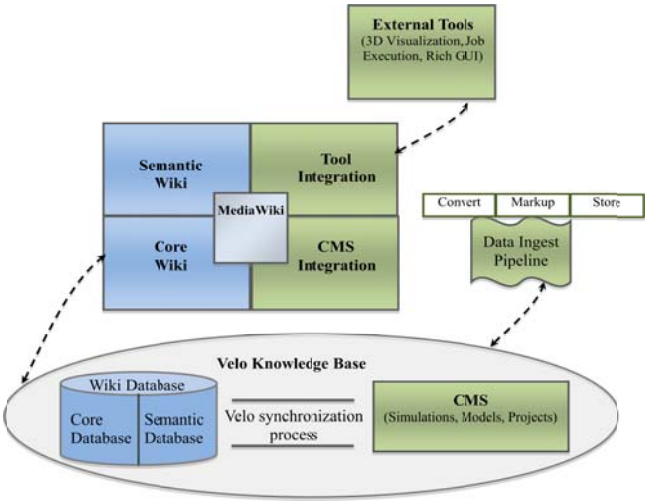

Figure 1. Building blocks of Velo architecture

MediaWiki provides a multi user environment for content generation, file and image upload, basic authentication, and version control of changes to pages. In Velo, we exploit these features to support the inherently iterative and exploratory nature of scientific modeling. Velo presents a web site structure based upon projects, models and data that can be customized to suit specific science domains. Using this structure, modelers can save metadata about projects and evolving model versions as wiki pages. This serves as documentation of the tacit knowledge for the modeling process, which can be shared with collaborators and/or used to generate reports for decision makers, regulatory bodies, and audits. Velo's page structure and user interface design is detailed in section 3.2.

Velo provides advanced search capabilities by exploiting the MediaWiki extension, Semantic MediaWiki (SMW)[2]. SMW enables the creation of arbitrary semantic tags that can be associated with content in any page, and supports querying the contents based on the tags created. It also provides interfaces to browse all semantic tags in a page and all pages with a specific tag. A query language similar to SQL can be used to retrieve pages with specific semantic tags, or a combination of search criteria. SMW's query capabilities are highly flexible, requiring no predefined schema. This enables users and tools to introduce new semantic tags as needed and build queries that retrieve tagged content.

While MediaWiki provides highly scalable management of the wiki page content using MySQL, all external content (e.g. linked files, images) is simply stored in the server's file system without any inherent organization. This makes it difficult to manage the large, structured collections of data that are generated during modeling projects.

Velo overcomes this by integrating MediaWiki with an open source content management system (CMS). The initial implementation is based on Apache Subversion (SVN)[3], a version control system that is robust and provides APIs for extension. Velo provides an integration infrastructure that synchronizes data file operations between MediaWiki and SVN. The implementation extends the MediaWiki database to save path information of SVN files, and utilizes MediaWiki and SVN provided programmatic hooks to achieve synchronization.

As an example, users can upload multiple versions of a file/folder into the SVN repository either through the Velo interface or using the SVN command line utilities. For every page uploaded or checked-in to SVN, an SVN post-commit hook makes a corresponding entry into the wiki database and creates a template or metadata page in the wiki. These pages are generated within a special namespace, WFS (wiki file system), and with the same name as the file's directory path. These pages contain summary information about the file/folder and provide a link to view/download the file. Velo provides interfaces for all basic file operations like upload, download, delete, rename, and move from all pages in the WFS namespace.

3.2 User Environment

The user interface (UI) design of Velo takes significant advantage of MediaWiki's skinning capabilities. This allows Velo to be presented using nearly any combination of interface design and development techniques. Thanks to the abstraction between the various application layers, the tailored UI allows for a custom look and layout without changing the underlying features and functionality of the environment.

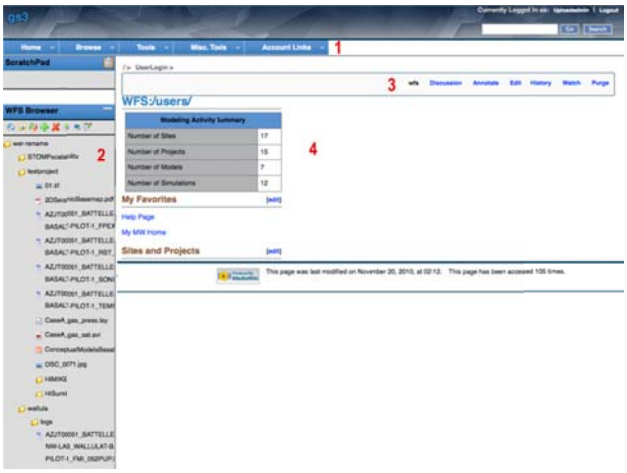

Figure 2. Basic Velo Skin

The Velo UI may be divided into four primary regions or areas as shown in Figure 2:

1. Tool Navigation – Velo integrates with various tools useful for the modeling process. These can be MediaWiki/SMW extensions or loosely coupled domain specific tools (see Section 3.4). Each tool available within and from Velo is dynamically included in this dropdown menu. When invoked, the tool will allow the user to select a data source, and then display the results in the content area.
2. File Browser – This gives asynchronous access to files in the underlying CMS. This area is resizable, giving the user control over how much space it takes up in their workspace. It also includes a *scratchpad* [28], or virtual collection space, allowing the user to collect multiple files from different directories as input for tools.

[2] http://semantic-mediawiki.org/wiki/Semantic_MediaWiki

[3] http://subversion.apache.org/

3. Wiki Functions – Access to standard wiki page functionalities such as annotation, revision history, discussion, and edit are presented inline with the content on each page.
4. Content Viewer – Project-, data- and image-based content are displayed in the largest region of the window by default, maximizing space to view tool results, analyze data, edit content, and collaborate.

This UI design leverages and extends the standard wiki look and feel, to give users a familiar workspace and experience. It is also easily modifiable and can be redesigned to provide multiple skins and customized views of the underlying Velo content.

3.3 Semantic Capabilities/Metadata Extraction

Metadata for file content in Velo is of two types, namely:

1. Generic information such as file size, owner, last modified timestamp and preview/thumbnails
2. Information specific to the file contents, such as keywords, geographic location, and included data types

By providing a metadata page for every file in the backend CMS, Velo provides a clean placeholder to present both types of metadata. This can help the user to quickly browse or query for information without having to open or download large numbers of potentially huge files.

When a user uploads a file, Velo stores it in the CMS and automatically extracts generic metadata information. It also provides an extensible framework for creating and invoking custom metadata extractors for different file types. Custom metadata extractors are subclasses of the Velo Extractor class, and they must register their associated file mimetype in a configuration file. Each time a file is uploaded, Velo automatically detects the mimetype of the file, invokes the appropriate metadata extractor, and creates a wiki metadata page with the contents returned from the extractor. Metadata pages conveniently enable the user to browse and query uploaded information without having to open and search through actual file content. Velo provides metadata extractors for common file types such as images and PDF, and we are currently developing extractors for Microsoft Office documents.

Figure 3 shows the result of Velo's PDF extractor. The wiki page contains a link to the actual file in the CMS and text extracted from the first page of the PDF file. A summary box is also generated that presents the occurrence of keywords in the doument. The keywords that the PDF extractor searches for are defined in a user extensible semantic dictionary, and the extractor also adds SMW format tags for the matching keywords to enable semantic searches on the document. The following shows a few example keywords based on a semantic dictionary for geological sequestration:

```
<category name="Rock Name">
  <key>Granite</key>
  <key>Granodiorite</key>
  <key>Rhyolite</key>
  <key>Sandstone</key>
  <key>Shale</key>
  <key>Siltstone</key>
  <key>Quartz</key>
  <key>Arkose</key>
</category>
```

This XML snippet defines eight keywords under the category *Rock Name*. Based on these definitions, the PDF extractor parses the uploaded file and finds, for example, the keyword *Sandstone* 17 times. It then creates a semantic property *Rock Name* with the value *Sandstone; 17* in the metadata page. that is generated for this document. Similar semantic tags are created for each matched keyword, and finally a template page is used to display the top matches in a summary table on the metadata page.

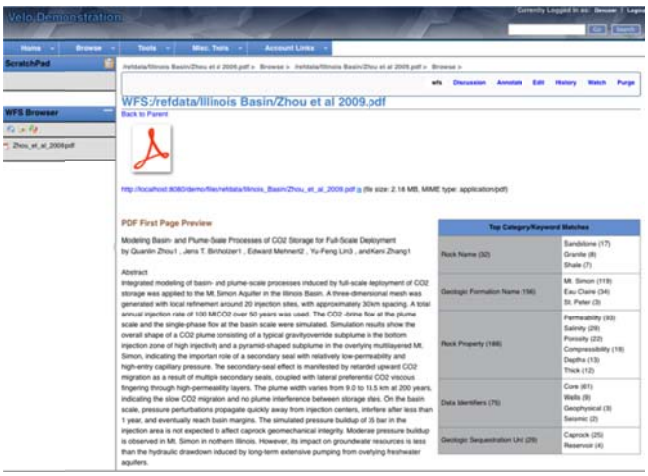

Figure 3. Metadata extracted from uploaded PDF

A metadata page for uploaded documents has several advantages:

1. Users see a summary of the contents of a file, allowing them to quickly identify useful information contained in the extracted text and a summary of keywords that are present in the file.
2. Users can add custom semantic annotations/tags, e.g. tacit knowledge related to the document, by simply editing the page and adding content of the format [[Property::Value]] or using the *Annotate* functionality of the SMWHalo extension[4].
3. Users can take advantage of SMW query features to search documents based on a combination of conditions. For example, a scientist working on geologic sequestration and looking for reference data can query for all documents with rock property *Permeability* mentioned more than ten times and with a geologic formation name of *Mt. Simon*.
4. By adding semantic markup to the metadata, Velo can take advantage of SMW extensions to present multiple views of the data. For example, data can be presented as points on a Timeline bar, as locations on a Google map, or as a thumbnail gallery.

For binary file types where no metadata can be extracted as well as for pages associated with special folders rather than uploaded files, users can create template pages with default contents and register that for the specific mimetype. In such cases, Velo creates a default metadata page with the contents of the template page. This default content serves as a starting point for user customization. For example, a default user's page could contain

[4] http://semanticweb.org/wiki/Halo_Extension

section headings 'My Favorites', 'My Projects', and a semantic query to generate a summary table of data in Velo. Users could edit the page to customize contents under individual section headings, customize the semantic query condition, or delete everything and insert custom content.

3.4 Tool Integration

Velo integrates a collection of tools with MediaWiki to provide a rich user environment for modeling and simulation. Some tools are closely integrated with the MediaWiki architecture and are implemented as MediaWiki extensions or hooks. Others are more loosely integrated where Velo only knows about the input and output format of the tool, and simply provides an interface to invoke the tool, treating the tool itself as a black box. Velo also provides loose integration with standalone tools that cannot be invoked from its infrastructure. We describe some examples of each tool type in the following subsections.

3.4.1 MW Integrated Tools

Velo leverages various open source plug-ins for MediaWiki by including them in its core architecture. We have also extended/modified some of them to work with the underlying CMS file structure. In particular, Velo integrates with:

1. SMW, an environment that provides rich query and visualization functionalities.
2. *Semantic Map* and *Map* extensions, used for querying and displaying data with geographic coordinates on Google maps.
3. *Semantic Gallery* and *ImageLink* extensions to display images in a gallery format. These are useful as scientific modeling and simulation generates large amounts of data in the form of images and plots.

In Velo, we have extended the SMW query engine to work with Velo's modified MediaWiki database to better support metadata pages in the WFS namespace. Since these pages represent the underlying hierarchical file system, we defined a new semantic query attribute, *scope,* to support file structure based queries. For any semantic query, Velo users can set an additional scope parameter with a value of local, recursive or global to restrict the search to files within the current (local) folder, current folder and all sub folders, or the entire SVN repository respectively.

Velo uses the HaloACL extension[5] to provide basic page level access control. We extended HaloACL to support inheritance of access control list (ACL) from the parent folder to child folders.

3.4.2 Black Box Integration

Velo supports loosely integrated tools, where the tool is not a part of MediaWiki source code or an extension. In such circumstances, Velo handles the input and output of the tool and invokes the tool using its interface.

Velo provides a framework for scientists to easily integrate their custom tools and invoke them from within Velo's wiki interface. A user with administrative privileges can create a MediaWiki *Special Page*[6] that can accept the user inputs for the tool, invoke the tool and redirect to the result page. Such tools are registered in a configuration file. Velo uses this configuration file to determine the point of integration in the wiki interface and the inputs required by the tool.

We are currently working on extending this framework so that the user can describe the tools' input/output specification, and the point of invocation from the Velo interface in an XML format. Velo would then use this to automatically create user input forms, validate inputs, invoke the tool, and redirect the output.

The following sections describe examples of loosely integrated or black box tools.

3.4.2.1 Job Launcher

One of the common needs across multiple scientific modeling domains is executing parallel numerical model simulations on remote high performance computing resources. Velo has implemented a job submission tool that creates a batch queue job control file for a variety of schedulers, submits the job, handles remote communication, and monitors the job's progress. Once a job completes, the specified result files are uploaded automatically into the Velo CMS. The job launcher is developed as a Java Web Start application that can be invoked from the Tools menu. Figure 4 shows the Velo Job Launcher interface.

Figure 4. Velo Job Launcher interface

3.4.2.2 Grid Viewer

The Grid Viewer is a visualization tool for data associated with a mesh and stored in a common gridded data format. The tool supports three suites of functions to facilitate visualization of a computational grid and associated attributes. The *Grid Control* functions provide options such as toggling on and off axes, bounding boxes and dual views; the *Slice Control* functions permit visualization of the interior of the volume; the *Cell-Render Control* functions control the visualization of attributes assigned to the individual cells and allows cutaway views using slider planes to visualize the interior domain. Figure 5 shows some of the visualization capabilities of the Grid Viewer.

3.4.3 External Tools

Scientists use many standalone software tools that are domain specific, proprietary and cannot be easily integrated with external software. The latter could be for reasons such as license restrictions, limited supported platforms, availability of APIs, and non-proprietary file formats. However, many of these are complex feature rich tools, which are critical to the scientists' everyday work. For such tools, Velo provides data integration through a check-out/check-in mechanism with the CMS. The user checks out the inputs required for the tool, uses the tool outside of the Velo framework, and then checks the output back into Velo. Velo in this case serves as a documentation tool and provenance tracker for this data. This provides users with the flexibility of using their existing tools, yet preserving the link between those data/processes performed within the Velo framework and those

---

[5] http://www.mediawiki.org/wiki/Extension:Halo_Access_Control_List

[6] http://www.mediawiki.org/wiki/Manual:Special_pages

performed outside the framework. Velo records basic information such as file owners and check-out/check-in timestamps but depends entirely on the users' documentation of the external processes to provide useful provenance information.

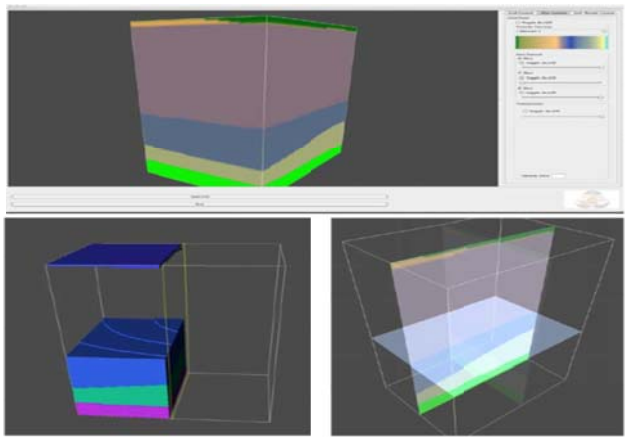

Figure 5. Visualizations supported by the Grid Viewer tool

4. Velo REALIZATION

Our initial deployment of Velo, the Geological Sequestration Software Suite ($GS^{3)}$, supports scientists working on geologic sequestration modeling projects. Geologic sequestration is a technology currently being developed to mitigate global warming through the long-term storage of greenhouse gases. For example, CO_2 emitted as a by-product of energy production can be captured and injected into subsurface reservoirs for permanent storage, thereby reducing emissions to the atmosphere.

Finding suitable subsurface (underground) sites for sequestration of greenhouse gases requires performing extensive studies that include modeling the site geology and the physical and chemical processes that determine the fate and transport of injected gases. Subsurface modeling is an iterative process that can often take months or even years to complete, and is continually revisited as new information is gathered. In addition, the work is performed not by a single scientist, but by interdisciplinary teams comprised of geologists, hydrologists, engineers, geochemists, and computational scientists who are unlikely to be co-located. As well as technical staff, there are project managers and others who may be reviewing or overseeing the work.

GS^3 is being used to aid in the organization and sharing of data and by facilitating collaboration within the teams of scientists charged with developing these complex models. The following subsections describe some of the key customizations and extensions to Velo developed for GS^3. It showcases how the framework makes it easy to customize and integrate with domain specific data and tools.

4.1 GS^3 UI Design and Custom Templates

For GS^3, a custom skin was developed using MediaWiki's skinning environment. This skin was created to give the GS^3 version of Velo a custom look and feel that clearly sets it apart from other Velo instances.

GS^3 organizes data in the CMS under top-level folders or directories named users, sites, projects and reference. The *users* directory serves as a scientist's personal development area. *sites* holds a collection of site-specific folders, each containing public (available to all users of GS^3) data. *projects* holds a collection of project directories that contain data related to a specific project/problem a single team is working on. The *reference* directory holds data that is not tied to any specific geologic site or project, but rather used as a general reference information.

GS^3 defines a semantic dictionary with various domain specific categories and keywords in the XML format specified in Section 3.3. It uses the PDF extractor to extract the first page of the document and produce a summary of keywords found in the entire document. This is extremely useful to the modelers as they often rely on reference documents in PDF format for certain data that are needed to develop a model but have not been collected at a particular study site. The metadata pages and advanced semantic query features greatly reduce their search set. As more users contribute reference documents to GS^3 and add semantic annotations to existing documents, the community benefits from the collective knowledge.

GS^3 uses Velo's mimetype specific metadata generation functionality to handle special directories/collections such as projects, models and simulations. These collections each have a specific meaning and thus interpretation when compared to normal file system directories/collections. GS^3 registers a new mimetype and a template page for each of these folder types. When a user creates one of these special folder types, GS^3 looks at the registered list of mimetypes and creates a metadata page using the corresponding template page. Figure 6 shows the metadata page that gets automatically generated for a new project folder. As projects are typically associated with a particular geographic location, the template page provides a summary table to show the project location along with a map. The user can input the location using a simple HTML form by clicking on the 'Edit with Form' link at the top. The HTML form is generated using the Semantic Forms extension.

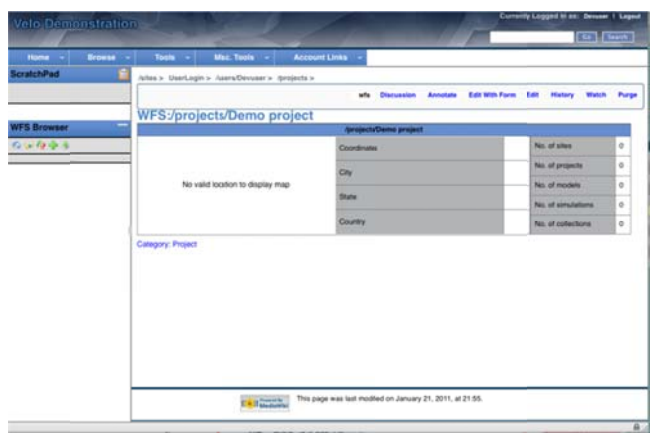

Figure 6. Default page created for any new GS^3 project

4.2 GS^3 Tools

The following are examples of tools created using the Velo tool integration framework for GS^3.

4.2.1 Well Log Metadata Extractor

The first step in the process of modeling the injection of CO_2 into the subsurface is to develop a conceptual understanding (or model) of the subsurface reservoir. An important type of data often used in this model development is geophysical wireline well log data. As a geophysical tool is lowered in the borehole, it records a variety of properties of the rocks and fluids to an

electronic data format creating a *well log*, typically in the industry standard Log ASCII Standard (LAS) file format. GS^3 uses the functionality described in Section 3.3 to register 'las' as a mimetype and provide a custom metadata extractor. When a .las file is uploaded, the system determines its mimetype and invokes its extractor. This extractor parses the file's header information to identify all of the recorded properties, the well's geographic location, and other useful metadata such as the log start depth and stop depth. The extractor also looks for a pre-configured list of properties and if found, creates a plot for each of these properties. These plots serve as a quick preview of the well data that is in the form of columns of numbers in the LAS file. Figure 7 shows a sample generated LAS metadata page with the plotted properties. It also shows all of the properties in the data file with an option to select and plot multiple properties for side-by-side comparison. The location of the well is also displayed on a Google map, along with a summary of data read from the header and a link to the actual file in the CMS.

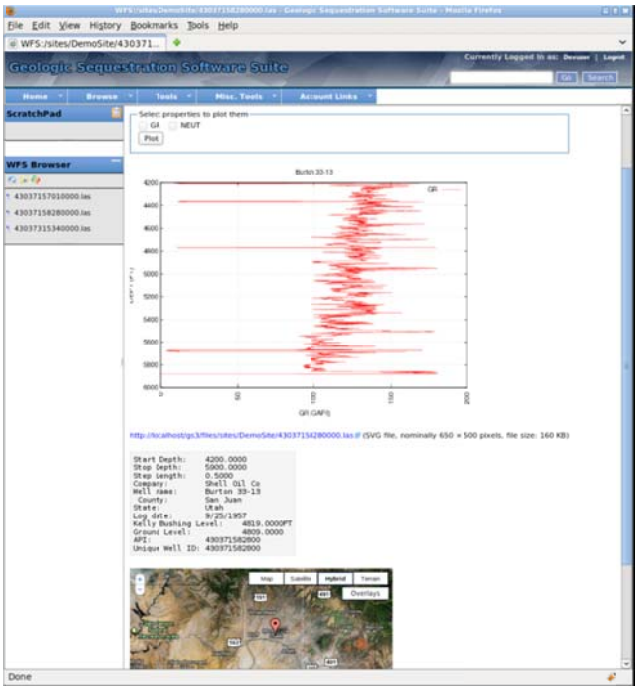

Figure 7. Metadata page generated for a LAS file

4.2.2 Well Log Browsing Tools

GS^3 provides some additional tools to support viewing data in multiple LAS files after upload and metadata extraction. One such tool is the *Graph LAS Properties*. This allows users to compare multiple well log files with respect to a single recorded property. The user can browse and select multiple LAS files from different directories, select a property common to all selected files, and create a plot of that property for all selected files side by side.

The *LASGallery* tool generates an image gallery of all the LAS files present in a given collection. This gallery uses the same plot presented in the LAS metadata pages. The *LASMap* tool uses the geographic coordinates extracted by the metadata extractor to create a Google map displaying the well locations. Both of these tools can use the *scope* attribute described in Section 3.4.1 to restrict the query results.

4.2.3 Data Extraction Tools

LAS files have data about various geologic properties recorded at different depths below the earth's surface. For a given problem, users are often interested in specific geologic properties at a specific depth range. GS^3 extends the SMW query functionality to create a new result format named 'las'. This result format accepts a depth range and the list of properties as additional inputs to a generic SMW query. SMW finds a list of LAS metadata pages based on given SMW query conditions. The GS^3 las format extension takes the resultant list as input; parses individual file contents, and extracts data corresponding to the specified properties and depth range. The extracted data is saved as another LAS file so that it can easily be downloaded and used as input for further processing.

4.2.4 Model Development and Documentation Support Tools

GS^3 also integrates with more loosely coupled "black box" tools as described in section 3.4.2. For example *Describe Conceptual Model* standardizes how conceptual model attributes for geologic sequestration are documented. This standardization facilitates comparison between models, structured queries, and summary overviews for managers and other stakeholders. Most importantly it serves as input for developing numerical models leading directly to simulation input files via additional GS^3 tools.

Describe Conceptual Model is implemented as a set of GUI worksheets that progressively collect more detailed information on the conceptual model in categories including *General Site Description*, *Infrastructure and Operations*, *Geologic Framework*, and *Processes and Properties*. The worksheets are invoked from the wiki via Java Web Start and are developed using an XML form layout specification in combination with the Jython scripting language. Jython is the Python scripting language implemented in Java yielding all the expected platform independence and Web interoperability of Java, but through a higher level of scripting language. Jython GUI components are implemented using a Python binding to Java Swing and AWT libraries allowing for robust and dynamic GUIs.

The Jython/XML form layout capability was designed as a flexible end-user extension capability to create UI based tools within Velo. The XML specification is tailored for collecting numeric input typical of scientific applications as well as for dynamically creating new fields, tabbed panes, and windows. Any behavior specific to the UI, such as enabling/disabling fields based on user settings for other fields, is easily implemented with Jython scripting. These are capabilities that are not easily supported using traditional Web-based forms. Figure 8 shows the *General Site Description* worksheet for *Describe Conceptual Model*.

When the data fields are completed, the current values are written to an XML *save state* file that is uploaded to the wiki via the CMS. Additionally, files that are referenced in field values are also themselves uploaded to the wiki. These are used for specifying auxiliary files that might include LAS well logs, GIS files, and others needed in the process of developing models, and ultimately simulation input files. A metadata page is generated from the XML *save state* file that provides both a formatted summary of the conceptual model and the semantic tags needed to perform queries. Finally, the XML *save state* file is used to

restore worksheet GUI fields when it is next invoked from the wiki allowing users to refine the model description throughout the development process.

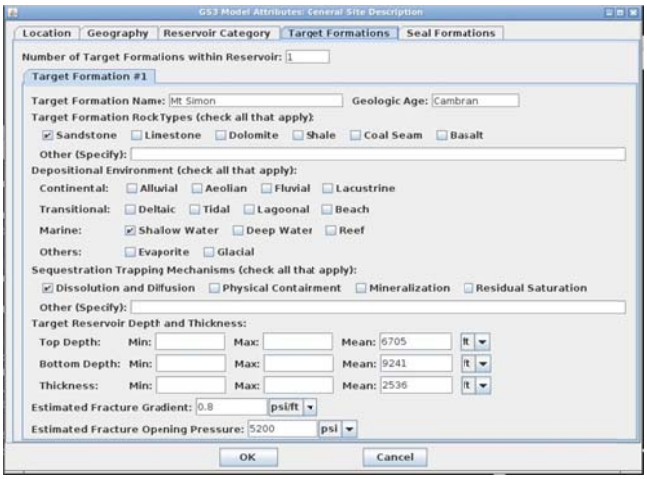

Figure 8. General Site Description Conceptual Model Worksheet

Another tool is *Estimate Plume Radius*, a standalone script that estimates the radius of the CO_2 plume given a specific set of inputs such as the mass of CO_2 injected, the depth of the injection, the temperature and the porosity of the formation. The result is a circle plotted on a Google map interface based on the location of the injection well and the estimated radius of the plume. Figure 9 shows a sample plume radius calculation.

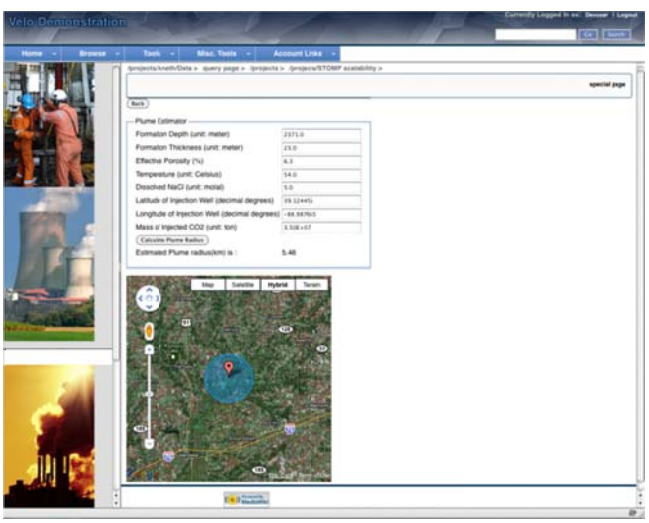

Figure 9. Plume Radius Estimator Tool

5. EVALUATION AND FURTHER WORK

The Velo framework is deployed in several instances of GS^3 that are supporting various user communities involved in the modeling of geologic sequestration sites. Velo is also a core framework component for the ASCEM (Advanced Simulation Capabilities for Environmental Modeling) User Platform, which is under active development. The framework is also being evaluated in prototypes for the CCSI (Carbon Capture and Simulation Initiative) Knowledge Management System, and the NEAMS (Nuclear Energy Advanced Modeling and Simulation) Data Management System.

The basic modeling and simulation lifecycle is common to all of these projects. The specific data types, tools, model inputs and simulation outputs vary, as do the high performance computing-based simulators. Based on our experience so far, we are able to accommodate a wide range of data types and integrate several different tools with Velo. This provides evidence of the flexibility and applicability of our architecture and confidence in its usefulness.

Our growing experience in working with scientists in different modeling and simulation domains has been highly effective in highlighting areas for extension and improvements for Velo. Some of the key features we are working on for our next release are as follows:

Provenance: An important element of quality assurance and assessment of simulation results is being able to understand the data and assumptions upon which the results are based. Velo's wiki page organization supports this to some extent, allowing navigation to view the inputs and outputs related to a simulation. However, we are investigating more rigorous, provenance [27] representation and navigation mechanisms that can provide much more extensive and powerful quality assurance processes.

Natural Language Processing: The current implementation of the Velo PDF Extractor only performs regular expression keyword matching on uploaded PDFs. A major enhancement would be to apply natural language processing to associate categories and properties that are matched with their associated quantitative (numbers) and qualitative (descriptions) values. This would facilitate a much more robust query capability to return useful results without users having to always refer to the underlying documents to discover the context for keyword matches.

6. CONCLUSIONS

As simulation and modeling projects increase in complexity, introducing rigorous knowledge management platforms is becoming a necessity. Our work in the Velo project is proving to be of utility to several scientific domains, and hence holds great promise in becoming the core of a next generation of flexible, scalable scientific knowledge management technologies.

We believe one of the major competitive advantages of Velo is its exploitation of robust, widely deployed open source technologies. This enables us to concentrate our efforts on integrating tools and data types relevant to scientists, and not invest on generic infrastructure level technologies. This approach is enabling us to build and deploy Velo environments for different domains at a much lower cost, while simultaneously creating an extensible collaborative platform for scientific knowledge management.

7. REFERENCES

[1] Bell, G.; Gray, J.; Szalay, A.; , "Petascale computational systems," *Computer* , vol.39, no.1, pp. 110- 112, Jan. 2006 doi: 10.1109/MC.2006.29

[2] J. Dongarra, P. Beckman, P. Aerts, F. Cappello, T. Lippert, S. Matsuoka, P. Messina, T. Moore, R. Stevens, A. Trefethen, and M. Valero, "The international exascale

software project: a call to cooperative action by the global high-performance community," Int. J. High Perform. Comput. Appl., vol. 23, no. 4, pp. 309–322, 2009.

[3] M. A. Heroux. Software challenges for extreme scale computing: Going from petascale to exascale systems. International Journal of High Performance Computing Applications, 23(4):437–439, 2009.

[4] D N Williams et al, Data management and analysis for the Earth System Grid, 2008 J. Phys.: Conf. Ser. 125 012072

[5] http://www.aquaveo.com/gms

[6] Yogesh L. Simmhan, Beth Plale, and Dennis Gannon. 2005. A survey of data provenance in e-science. *SIGMOD Rec.* 34, 3 (September 2005), 31-36.

[7] Gil, Y.; Deelman, E.; Ellisman, M.; Fahringer, T.; Fox, G.; Gannon, D.; Goble, C.; Livny, M.; Moreau, L.; Myers, J.; , "Examining the Challenges of Scientific Workflows," *Computer* , vol.40, no.12, pp.24-32, Dec. 2007.

[8] Reagan Moore. 2000. Data Management Systems for Scientific Applications. In *Proceedings of the IFIP TC2/WG2.5 Working Conference on the Architecture of Scientific Software*, Ronald F. Boisvert and Ping Tak Peter Tang (Eds.). Kluwer, B.V., Deventer, The Netherlands, The Netherlands, 273-284.

[9] M. Earl, Knowledge management strategies: toward a taxonomy, Journal of Management Information Systems 18(1), 2001, pp. 215–223.

[10] Stefan Bitzer, Stefan Ramroth, Matthias Schumann, "Mashups as an Architecture for Knowledge Management Systems," Hawaii International Conference on System Sciences, pp. 1-10, 42nd Hawaii International Conference on System Sciences, 2009.

[11] V. Sambamurthy, and M. Subramani, Information technologies and knowledge management, MIS Quarterly Vol. 29, No. 1, 2005, pp. 1-7.

[12] T. H. Davenport, D. W. De Long, and M. C. Beers, Successful knowledge management projects, in the Knowledge Management Yearbook 1999-2000, 2000, pp. 98-107.

[13] K.L. Schuchardt, B.T. Didier, G.D. Black, "ECCE - A Problem-Solving Environment's Evolution Toward Grid Services and a Web Architecture," Concurrency and Computation: Practice and Experience 14, 1221-1239, 2002.

[14] Schwidder, J.; Talbott, T.; Myers, J., "Bootstrapping to a semantic grid", Cluster Computing and the Grid, 2005. CCGrid 2005. IEEE International Symposium on Cluster Computing, vol.1, no., pp. 176-181 Vol. 1, 9-12 May 2005.

[15] Shah, A.R.; Singhal, M.; Gibson, T.D.; Sivaramakrishnan, C.; Waters, K.M.; Gorton, I., "An Extensible, Scalable Architecture for Managing Bioinformatics Data and Analyses," IEEE Fourth International Conference on e-Science, vol., no., pp.190-197, 7-12 Dec. 2008.

[16] Christian Hirsch, John Hosking, John Grundy, Tim Chaffe, David MacDonald, Yuriy Halytskyy, "The Visual Wiki: A New Metaphor for Knowledge Access and Management," pp. 1-10, 42nd Hawaii International Conference on System Sciences, 2009.

[17] Mika, P., Flink: Semantic Web technology for the extraction and analysis of social networks. Journal of Web Semantics, 2005. 3(2-3): p. 211-223.

[18] http://websrv.cs.umt.edu/isis/index.php/Main_Page

[19] http://oceans11.lanl.gov/trac/CISM

[20] S. Jeschke, M.Wilke, N. Natho, O. Pfeiffer, "Knowledge Base Concepts in the KEA System Combined with Social Networking Techniques," pp. 1-9, 42nd Hawaii International Conference on System Sciences, 2009. IEEE

[21] S. Bitzer, S. Ramroth, M.Schumann, "Mashups as an Architecture for Knowledge Management Systems," pp. 1-10, 42nd Hawaii International Conference on System Sciences, 2009. IEEE

[22] M. Lindvall, I. Rus, and S. Sinha, "Software systems support for knowledge management," in Journal of knowledge management, Vol. 7, No. 5, 2003, pp. 137-150.

[23] Damian Flannery, Brian Matthews, Tom Griffin, Juan Bicarregui, Michael Gleaves, Laurent Lerusse, Roger Downing, Alun Ashton, Shoaib Sufi, Glen Drinkwater, Kerstin Kleese, "ICAT: Integrating Data Infrastructure for Facilities Based Science," e-Science and Grid Computing, International Conference on, pp. 201-207, 2009 Fifth IEEE International Conference on e-Science, 2009

[24] Sobha Mathew Tharayil, Ted Kalbfleisch, Anup Kumar, "Service-Oriented Laboratory Information Management System for Life Sciences Research," Services Computing, IEEE International Conference on, pp. 621-627, IEEE International Conference on Services Computing (SCC 2007), 2007.

[25] Kiebel, G. R., Auberry, K. J., Jaitly, N., Clark, D. A., Monroe, M. E., Peterson, E. S., Tolić, N., Anderson, G. A. and Smith, R. D. (2006), PRISM: A data management system for high-throughput proteomics. PROTEOMICS, 6: 1783–1790.

[26] Chaitanya Baru, Reagan Moore, Arcot Rajasekar, and Michael Wan. 1998. The SDSC storage resource broker. In *Proceedings of the 1998 conference of the Centre for Advanced Studies on Collaborative research* (CASCON '98), Stephen A. MacKay and J. Howard Johnson (Eds.). IBM Press

[27] Yogesh L. Simmhan, Beth Plale, and Dennis Gannon. 2005. A survey of data provenance in e-science. *SIGMOD Rec.* 34, 3 (September 2005), 31-36.

[28] Zimmerman, J., Rivard, K., Hargraves, I., Tomasic, A., Mohnkern, K. User-created forms as an effective method of human-agent communication. Proceedings of the 27th international conference on Human factors in computing systems, p. 1869-1878, 2009, ACM New York, NY, USA.

[29] Ian Gorton, Gary D. Black, Karen L. Schuchardt, Chandrika Sivaramakrishnan, Signe K. Wurstner, Peter Hui, "GS3: A Knowledge Management Architecture for Collaborative Geologic Sequestration Modeling," Hawaii International Conference on System Sciences, pp. 1-10, 2010 43rd Hawaii International Conference on System Sciences, 2010.