WEB-ENABLED TOOLKIT FOR DATA INTEROPERABILITY SUPPORT

Jigarkumar Patel, Sohei Okamoto, Sergiu M. Dascalu, Frederick C. Harris, Jr.

Department of Computer Science & Engineering
University of Nevada Reno
Reno, NV, 89557, USA
{jspatel, okamoto, dascalus, fredh}@cse.unr.edu

Abstract

To address some of the challenges related to data interoperability in scientific modeling and simulation we propose a web-enabled software application (software toolkit) that dynamically generates data processing tools (data processors) capable of performing conversions and other operations on data based on user-defined specifications and customizations. The proposed application is designed to store all new user-created data structures and mappings for reuse and sharing purposes. It is also designed to generate reusable source code and executable binary files for easy and straightforward use by the scientists, thus saving them a substantial amount of data processing time. This paper presents the software architecture of the proposed web-enabled toolkit, provides details of the toolkit's modes of operation and web user interface, and outlines directions of future work.

1. INTRODUCTION

Usually a large dataset is considered a gold mine for the scientists, but often it is hard to find the desired subsets of data from this dataset. Depending on the dataset's type and size researchers employ various techniques to get subsets of data of interest, including time consuming manual filtering or writing a script or an application that could be run by a computer. This process prolongs the core research process and also leaves room for errors. This paper focuses on data processing issues related to geospatial data. These issues can largely be divided into three main categories. The actual practical problem encountered by a scientist could be the result of a combination of these issues.

1.1 Data Storage Formats

Researchers acquire data from various public repositories and other data sources. The datasets of interest often come in different file formats. Sometimes datasets are also a byproduct of model simulations. The same dataset can be represented differently in different files. Figure 1 illustrates the same data in some of the more popular human readable file formats. Conversions between various file formats typically present challenges to the researchers. Many times they have to spend a significant amount of time writing code to handle file format conversions and the same process gets repeated in other research groups.

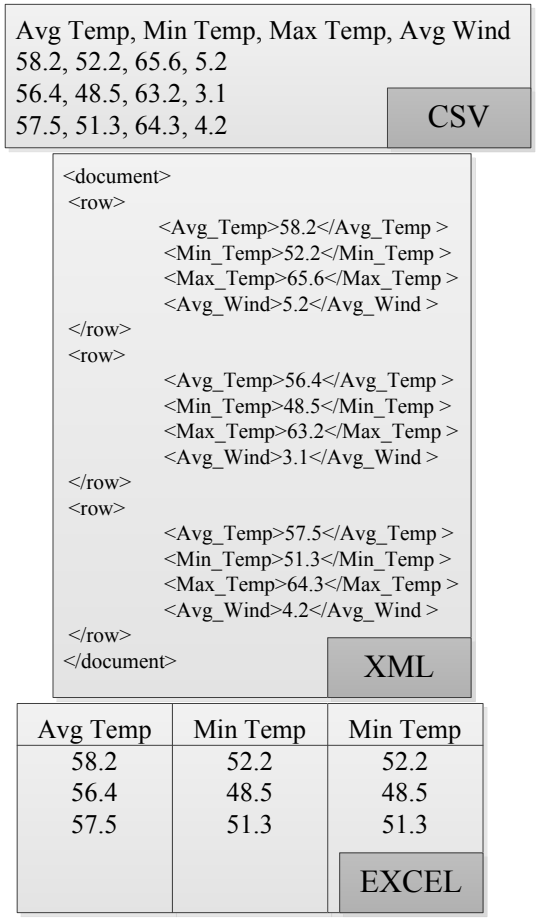

Figure 1: Data represented in different file formats

1.2 Data Filtering, Merging, Sorting and Grouping

Many time series data repositories store large amounts of data spanning over many years and consisting of many records. Some repositories also have live data streaming. Besides time series data, many geospatial data files are quite large, covering large geographical areas. Researchers often need to collect subsets of data from several sources. With online public repository access tools it is easier to download the desired time series data but

this is not possible with locally stored time series and geospatial data. Model coupling often requires combining data sources to create necessary datasets to use with the models. The data filtering and merging problem is illustrated in Figure 2, in which rectangles represent data files and circles represent individual records in the files. Circles with the same colors are an indication of matching records based on a user provided filtering condition. The top and bottom files represent two distinct dataset sources. The middle rectangle is a resulting merged data file obtained by applying filtering conditions to the source datasets. The overall process is further complicated by different file formats of source and destination. Adding grouping and sorting capabilities for data only makes the problem even more challenging.

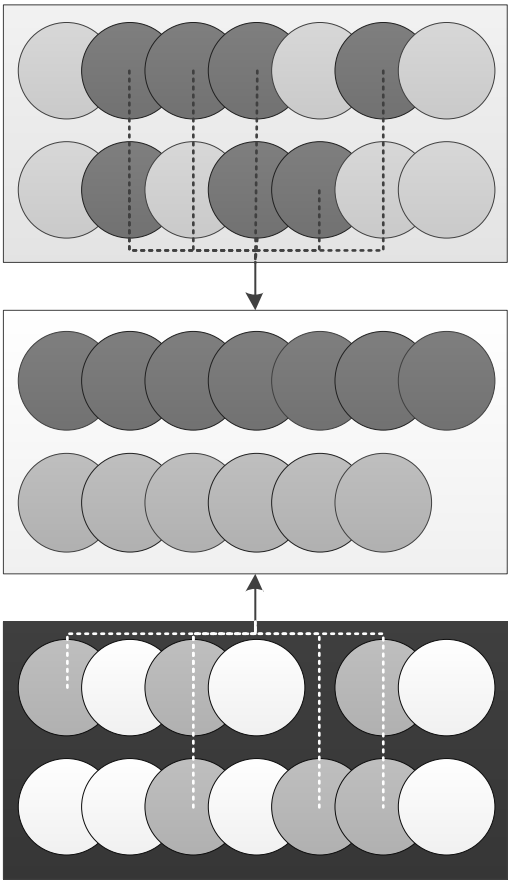

Figure 2: Data filtering and merging process illustration

1.3 Data Scaling Challenges

Various environmental science models use geospatial data as part of their input and produce geospatial datasets as their output. These models often operate at different geological scales hence they require datasets at different scales and frequency. Scaling datasets is a complex challenge, as illustrated in Figure 3 and Figure 4. In both figures, a black dot is an actual value for a climate variable and a rectangular area represents a grid cell that is part of a larger geographical area (grid). Down sampling data of a grid cell to smaller size cells is largely dependent on the geological location and type of parameters and models researchers are working with. Similarly, many models operate at different time scales and down sampling time scaled data also depends on climate variables and models involved in the process. Climate models generally operate on a larger size grid with larger cell areas whereas hydrology models operate on a smaller size grid with smaller cell areas [1]. Thus, hydrological models can use the output results of climate model simulations but not without down scaling the output data. The down sampling issue becomes even more complex with rapidly changing geographical terrain such varied combinations of mountains and valleys.

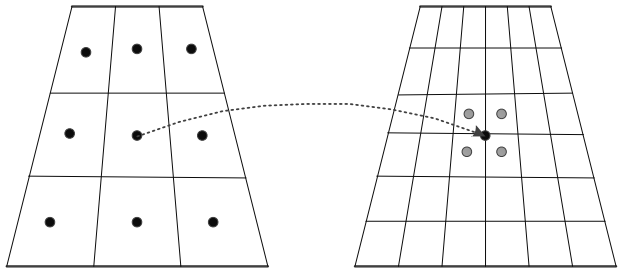

Figure 3: Geospatial data grid down sampling

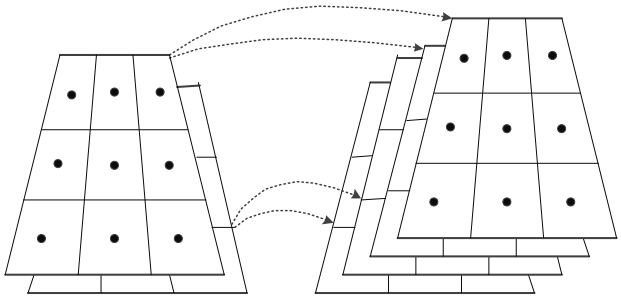

Figure 4: Geospatial time data down sampling

2. RELATED WORK

Researchers from different domains have access to various public repositories to obtain data. For example, the National Oceanic and Atmospheric Administration [2] and other similar repositories offer web-based interfaces to download data using map and selection tools. However the users do not have control over the output climate variables and do not have many choices regarding the file format of the downloaded data. For data processing and model simulation activities, researchers can use scientific workflow applications such as Kepler [3] [4]. Kepler enables users to select and connect pertinent analysis or computational components and data sources to create an executable representation (scenario) of the steps required to generate results. Kepler is a powerful environment with

many capabilities but has a steep learning curve and the workflows produced using it require it to run, hence making the newly created solutions dependent on Kepler. There are also other tools [5] [6] [7] [8] available to process locally stored data but they are bound to certain specific data types only and generally do not offer easy ways for extension. In comparison, the solution proposed in this paper offers a new and flexible approach aimed at handling many data interoperability challenges not directly addressed by others.

3. DESIGN OF THE PROPOSED SOLUTION

To provide tool support for addressing data inter-operability issues, we are developing a web application whose high level design can be represented using subsystems [9], as shown in Figure 5. Each subsystem consists of a set of functions included to fulfill the application's requirements. Each subsystem is explained in further detail in the following subsections.

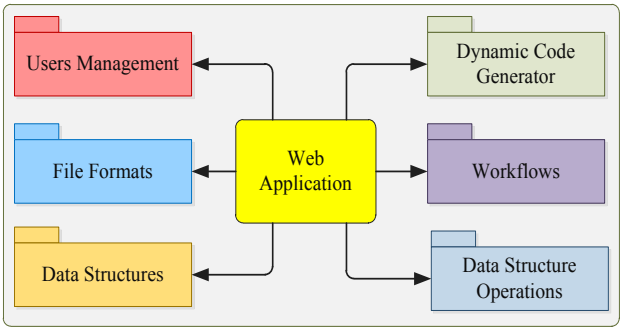

Figure 5: Subsystems of proposed solution

3.1 Subsystems

3.1.1 Users Management

As its name suggests, this subsystem manages all processing aspects pertaining to users, including user registration, login, logout, profile update, password change, password reset, and ownership of the content. Administrators of the web application can also manage users by creating new users directly without registration.

3.1.2 File Formats

This subsystem is designed to facilitate support for reading and writing data in different formats. Only an application developer can add support for a new file format as it requires changes in the source code and recompilation of the software. Web applications users will be able to select appropriate file formats during the workflow creation process. This is a critical subsystem because supported file formats determine which models can be used for the purpose of data processing tools' generation.

3.1.3 Data Structures

This subsystem can be considered the fundamental building block in the model coupling process. This subsystem is designed to allow users to define and manage data structures assembled from a variety of primitive data types such as number, text, date, time, and Boolean. Support for complex data structures will be included in this subsystem in future releases.

3.1.4 Data Structure Operations

This subsystem will enable users to define grouping, sorting, and logical filtering conditions on particular data structures. Grouping and filtering conditions are defined on columns (data structure components) whereas logical filtering conditions are defined on column values using conditional operators and functions.

3.1.5 Workflows

This subsystem depends on all other subsystems except the code generator subsystem. It will provide a user interface to accomplish the ultimate tasks of data processing tools' generation. Users will be able to create complex workflows by simply using drag and drop interactions. As a part of the process, users will define required data structures, associated operations, and file formats. The subsystem will also be able to generate reusable workflows that will relieve researchers from time-consuming repeatable data processing activities and could eliminate human errors that might be introduced in the manual process.

3.1.6 Dynamic Code Generator

This subsystem will generate a wrapper source code that will allow the execution of user created workflows. The generated source code will be based on data structures, data structure conditions, and file formats defined or used in the process of workflow creation. The subsystem will also generate an executable binary file from the generated source code by compiling it on the fly. The users will be able to download the source code and its associated binaries with dependent class libraries. Furthermore, the users will be capable to execute the downloaded workflow binaries on their local computers, without any user intervention after initiating the execution process. Hence, the proposed approach will provide a complete start-to-finish data processing tool that can be used by researchers without writing any code.

3.2 Software Architecture Patterns

In software engineering, developers can choose from many software architecture patterns, as over the years many architecture patterns have evolved and matured. The choice of architecture patterns largely depends on the project requirements and the developers' experience. Furthermore, it is possible to deploy multiple architecture

patterns [10] in the same project. The proposed solution can be divided essentially into two parts: a web application for user interaction, and a class library to implement the actual tasks. Both parts of the solution use different architecture patterns to leverage the software engineering principles of flexibility and reliability. The two patterns employed also come with their specific software design benefits.

3.2.1 Model View Controller Pattern

The proposed solution is being developed partially based on the Model-View-Controller (MVC) architecture principles. More specifically, the solution is developed using the ASP.Net MVC 3 [11] [12] [13] framework and is coded in the Visual C# programming language. Although the solution follows to some extent the MVC framework architecture it also uses the Model-View-ViewModel (MVVM) architecture pattern to break down the functionality into logical sections and thus create a data independent layered architecture.

The Model-View-Controller (MVC) is a software architecture pattern that separates data access, domain logic, and user interface from each other. The MVC can be graphically represented as shown in Figure 6. The MVVM pattern adds an additional layer of data representation, as depicted in Figure 7. In the figures, the solid line represents a direct association whereas the dashed line represents an indirect association.

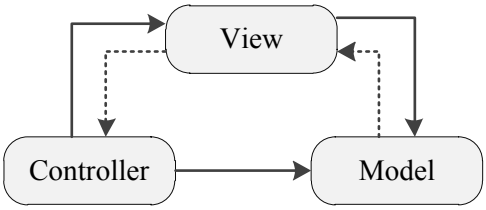

Figure 6: Model-View-Controller (MVC) Overview

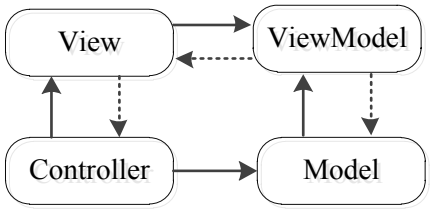

Figure 7: Model-View-ViewModel (MVVM) overview

In the MVC pattern, the Model represents the data and the associated domain logic of the application. It is also responsible for providing responses to requests and for being aware of the state of data. It can also be seen as an object model for data entities. The View translates the model into forms suitable for user display and interaction. The View generally is not aware of the domain logic and only takes care of rendering the given data. It passes all user requests and other events to the Controller. The Controller receives user input and requests via form submissions or events. It transmits the submitted forms and requested actions to the Model and after receiving responses from the Model it sends data back to the View for user presentation and interaction.

The proposed solution also employs the MVVM architecture pattern. In this pattern the View-Model is responsible for exposing the data objects resident in the Model in ways that allow these objects to be easily managed and used. Both MVC and MVVM emphasize separation of concerns in software design, implementation and testing, thus enabling developers to create higher quality code.

3.2.2 N-tier Architecture and the Repository Pattern

To overcome certain limitations of the MVC and MVVM patterns, the proposed solution also uses a multi-tier architecture (or N-tier architecture) for the class library part of the web-enabled toolkit's design. The N-tier architecture separates processes in three categories: presentation, application processing, and data management. Services defined by interfaces handle data requests between a user and a database. This architecture enables developers to create flexible and reusable applications; specifically, by structuring an application into tiers, developers have only to change (or add) a specific layer, without the need of rewriting the entire application. The overall software architecture of the web-enabled toolkit proposed in this paper is a combination of the N-tier architecture and the repository software pattern. Especially for applications with complex domain models, a repository can serve as a useful mediator between the domain and data mapping layers, acting as a domain object collection residing in the memory. For effective data access layer, the proposed solution also uses the Object Relational Mapping (ORM) framework, called the Entity Framework by Microsoft [14].

4. USER INTERACTION AND USER INTERFACE

In the first step of the process, the proposed toolkit allows the users to define various data structures composed of predefined primitive data types, as illustrated in the activity diagram shown in Figure 8.

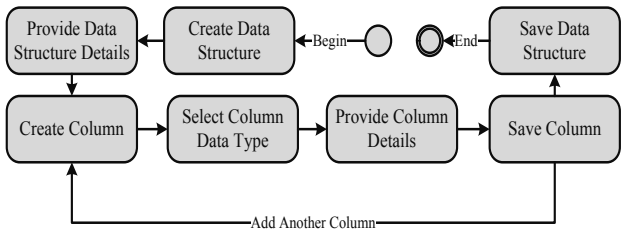

Figure 8: Create new data structure process

A web interface snapshot pertaining to the same process is shown in Figure 9. The resulting sample data structure is presented in Figure 10. The next step of the process is to define data operations on existing data structures by defining grouping, filtering, and sorting conditions on structure components (columns), as illustrated in Figure 11. Interface screenshots representative of this process are shown in Figure 12. In the next step of the process the users can create a desired workflow by selecting the input file format, the specific data structure involved, the data operations that are applied on the specified data structure, and the required output file format. This process is illustrated in Figure 13. If desired, the users can also select appropriate downscaling functions. Finally, the users will be able to download the source code and executable binaries generated based on the workflow created.

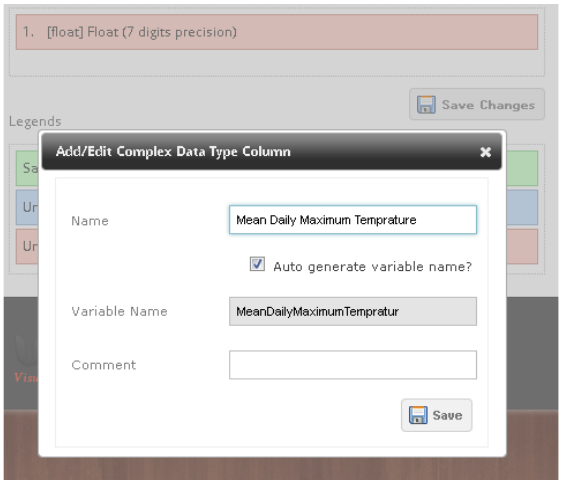

Figure 9: Interface to add a new data structure

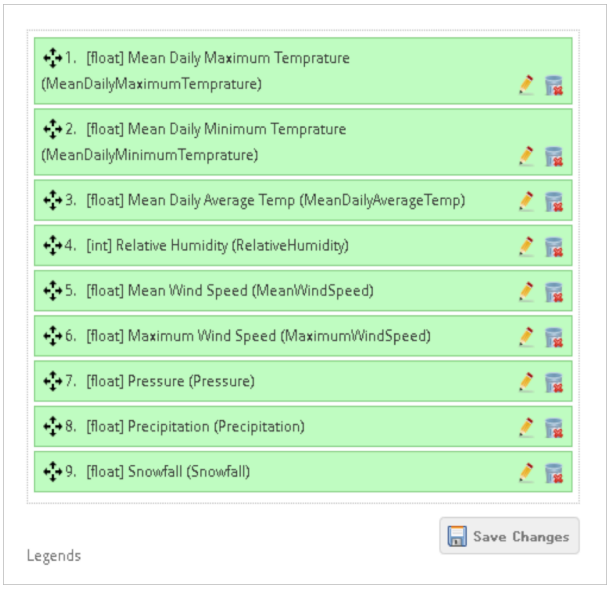

Figure 10: Data structure created via the web interface

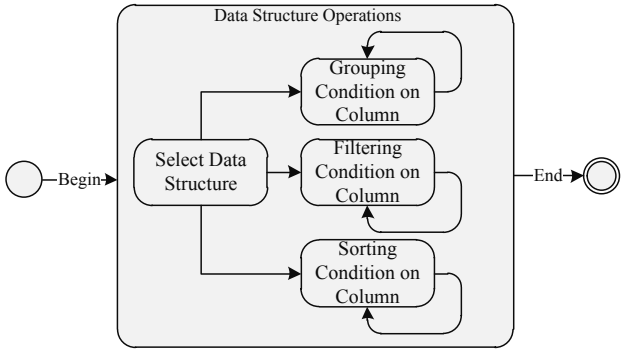

Figure 11: Defining operations on data structures

5. CONCLUSIONS AND FUTURE WORK

Data processing in general and data interoperability in particular raise complex challenges for which there are no "one size fits all" solutions. The key objective of the work presented in this paper has been to provide enhanced support for data interoperability (with particular application to climate change research). To the best of our knowledge, current solutions do not tackle data interoperability challenges via a web-based approach. With our proposed web-enabled toolkit, dynamically generated data processors will allow more efficient testing of research hypotheses and provide a basis for addressing more complex model and data interoperability problems through contributions from the research and development community.

The web-enabled application presented in this paper aims to provide a new way for addressing data interoperability issues. At this point in time, there are many possibilities for improvements and directions of future work. For example, one such direction is to auto-detect data structures by reading file headers and thus reduce end-user work. At this stage source code generation is based on internal templates which generate code in the Visual C#/Mono programming language. In the future, it is possible to add templates for other programming languages to meet additional requirements of the researchers. Furthermore, we plan to add data downscaling functions to our web-enabled toolkit. Also, besides local files, we would like to add support for various other data sources such as web services and relational databases. The workflow generation process can also be expanded to include additional data processing and computational activities and thus support defining and executing more complex model coupling scenarios.

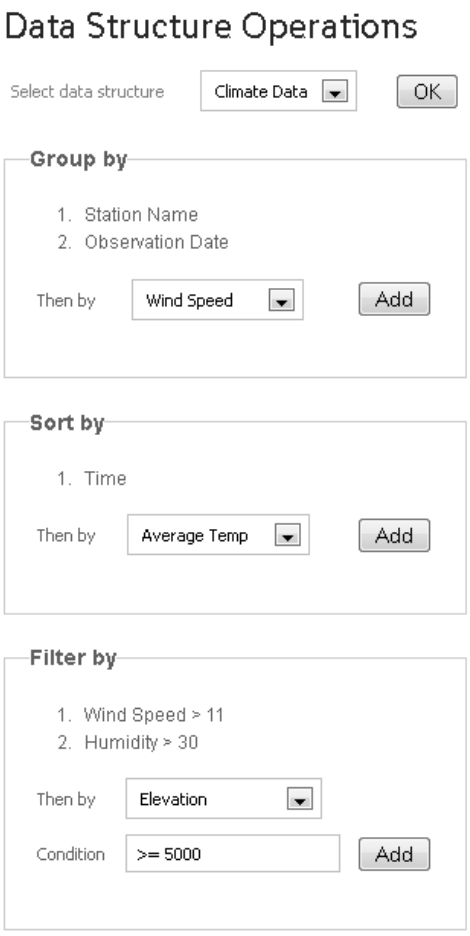

Figure 12: Sample data structure operations

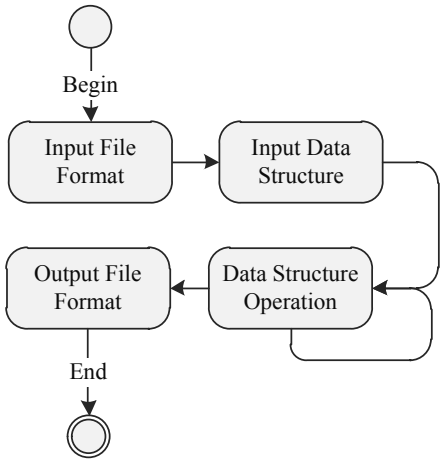

Figure 13: The core workflow creation process

Acknowledgments

This work was made possible through the support provided by the National Science Foundation under Cooperative Agreements No. EPS-0814372 and No. EPS-0919123.

6. REFERENCES

[1] A. Bernea, G. Delrieu, J. D. Creutin, and C. Obled, "Temporal and spatial resolution of rainfall measurements required for urban hydrology," *Journal of Hydrology*, vol. 299, no. 3-4, pp. 166-179, December 2004.

[2] National Oceanic and Atmospheric Administration. (2012, April) NOAA Climate Services. [Online]. Available at: www.climate.gov

[3] San Diego Supercomputing Center, "Kepler: an extensible system for design and execution of scientific workflows," in *Proceedings of the 16th Intl. Conference on Scientific and Statistical Database Management*, 2004, pp. 423-424.

[4] The Kepler Project (2012, April) The Kepler Project. [Online]. Available at: https://kepler-project.org/

[5] D. Kavan and P. Man, "MSTools—Web based application for visualization and presentation of HXMS data," *International Journal of Mass Spectrometry*, vol. 302, no. 1-3, pp. 53-58, 2011.

[6] M. Nilsson, "The DOSY Toolbox: A new tool for processing PFG NMR diffusion data," *Journal of Magnetic Resonance*, vol. 200, no. 2, pp. 296-302, October 2009.

[7] M. Waldhor and E. Appel, "Intersections of remanence small circles: new tools to improve data processing and interpretation in palaeomagnetism," *Geophysical Journal International*, vol. 166, no. 1, pp. 33-45, July 2006.

[8] C. Schwager, U. Wirkner, A. Abdollahi, and P. Huber, "TableButler – a Windows based tool for processing large data tables generated with high-throughput methods," *BMC Bioinformatics*, vol. 10, pp. 235-243, 2009.

[9] I. Sommerville, *Software Engineering*, 9th edition, Addison-Wesley, 2010.

[10] M. Fowler, *Patterns of Enterprise Application Architecture*, Addison-Wesley Professional, 2002.

[11] Microsoft ASP.NET MVC (2012, April) MVC: The official Microsoft ASP.NET Site. [Online]. Available at: http://www.asp.net/mvc

[12] A. Freeman and S. Sanderson, *Pro ASP.NET MVC 3 Framework*, 3rd edition, Apress, 2011.

[13] J. Galloway, P. Haack, B. Wilson, and S. Allen, *Professional ASP.NET MVC 3*, Wrox, 2011.

[14] J. Lerman and R. Miller, *Programming Entity Framework: Code First*. O'Reilly Media, 2011.