Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 108C (2017) 2090–2099

Procedia Computer Science

International Conference on Computational Science, ICCS 2017, 12-14 June 2017, Zurich, Switzerland

High-Level Toolset For Comprehensive Visual Data Analysis and Model Validation

Konstantin Ryabinin[1] and Svetlana Chuprina[1]

[1]*Perm State University, Bukireva Str. 15, 614990, Perm, Russia*

kostya.ryabinin@gmail.com, chuprinas@inbox.ru

Abstract

The paper is devoted to the new method of high-level scientific visualization, comprehensive visual analysis and model validation tools development using new version of client-server scientific visualization system SciVi as an example. The distinctive features of the methods implemented are ontology-based automated adaptation to third-party data sources from various application domains and to specifics of the visualization problems as well as multiplatform portability of the software solution. High-level tools for semantic filtering of the rendered data are presented. These tools improve visual analytics capabilities of SciVi enabling to validate solvers' or/and data sources' models in more comprehensive form and to reduce uncertainties due to the explicit representation of hidden features of data.

Peer-review under responsibility of the scientific committee of the International Conference on Computational Science

Keywords: scientific visualization tools, ontology engineering, data filtering, visual analytics, model uncertainty reduction

1 Introduction

While scientific visualization is widely used to analyze and interpret various data, there are still issues to be tackled in this field. The state-of-the-art approaches to render scientifically valuable data often concentrate on the visualization algorithms avoiding the question of retrieving and preparing the data. Popular scientific visualization systems like ParaView, KiwiViewer, VizIt, TecPlot, Avizo, etc. require the input data in the standard formats. Although there are a lot of formats supported, they still cannot cover every single permutation of the input data and suite the needs of every researcher. This means, if the scientist works with some data generated by software or hardware solver, or retrieved them from data storage, he/she can encounter compatibility problems when the particular data source does not conform to the format required by the visualizer.

The traditional solution of compatibility problem is either to build the intermediate conversion software or to change the source code of the used solver to make its input match the desired format.

1877-0509
Peer-review under responsibility of the scientific committee of the International Conference on Computational Science
10.1016/j.procs.2017.05.050

Both cases require the scientist to have related skills in programming or to hire developers. As a result, that may delay the research process and hamper the attainment of research goals. We discussed these and others problems of traditional scientific visualization systems in details in our previous papers (see, for example, (Ryabinin K. & Chuprina S., Adaptive Scientific Visualization System for Desktop Computers and Mobile Devices, 2013)).

To solve the compatibility problem in a more smart way, we propose a high-level adaptation mechanism based on model-driven architecture, which is implemented within scientific visualization system SciVi. That allows fine-tuning of the visualization system to make it compatible with third-party data sources without changing their format. Technically the adaptation mechanism proposed provides high-level tools that automate data converters building and allow automatically reconfiguring the visualization algorithms to tie them to the required data structures.

Moreover, the proposed adaptation mechanism provides user with high-level editor of the visualization system's rendering pipeline. This editor allows the user to set up preprocessing of the data, which are to be rendered, enabling, for example, to filter out irrelevant values, change scale from linear to logarithmic, etc.

If the data source is a software solver generating the data to be visualized in runtime, adaptation mechanism proposed allows automatically create the graphical user interface (GUI) to steer that solver, enabling the feedback from the visualization system. Using the interface created, user can restart the solvers' calculations; change input parameters and so on.

Model-driven architecture ensures easy extensibility of the visualization system, including the ability to add new data processing and rendering algorithms (or modify the existing ones) without changing the source code of the system's core. It is enough to modify the underlying model to change the system's behavior.

The other problem of popular modern scientific visualization systems is a lack of multiplatform portability. But the growing number of platforms widely used today, including mobile devices, makes this problem crucial. If the system can run on mobile devices as good as on desktop computers, it can be used both during the expedition and in the laboratory, providing the scientist with continuous working process with no need to switch between different software tools.

To ensure multiplatform portability we propose client-server architecture of visualization system with two types of clients: thick and thin one. Thick client is a native multiplatform application built for Windows, GNU / Linux, macOS, iOS and Android. Thin client is a cross-browser Web application capable to run in Firefox, Chrome, Safari, Opera and Internet Explorer (including their mobile versions).

In this paper we specify the concept of our approach to organize the feature-reach high-level toolset for scientific visualization and visual data analysis, as well as describe technical details of its implementation within SciVi.

2 Previous Work

Our previous investigations in scientific visualization area showed the following main problems the researchers may encounter (Ryabinin K. & Chuprina S., Development of Ontology-Based Multiplatform Adaptive Scientific Visualization System, 2015):

1. Lack of the high-level mechanisms to adapt the scientific visualization system to the third-party data sources and specifics of the application domain without any changing in data source representation.
2. Inability to set up the feedback with the solver generating data without changing the solver's or visualizer's source code.

3. Deficiency of high-level means to extend the graphical capabilities of the visualization system.
4. Absence of the portable software that could run both on desktop computers and mobile devices.

To alleviate all these problems we propose an approach to build model-driven scientific visualization systems based on ontology engineering methods. We implemented this approach in multiplatform scientific visualization system named SciVi. The detailed description of ontologies' role in the adaptation process of software components to the specifics of concrete scientific visualization tasks including a running example is available in our previous paper (Ryabinin K. & Chuprina S., Development of Ontology-Based Multiplatform Adaptive Scientific Visualization System, 2015).

SciVi consists of server and client parts. The server is responsible to obtain data from data source, and the client plays a role of the interface with the user. Visualization process is adaptively distributed between client and server to ensure the best quality and highest interactivity in particular computational environment.

The key features of SciVi are:

1. 2D and 3D rendering (in prospect, high-dimensional rendering can be implemented).
2. Objects' and scenes' repository extensibility.
3. High-level ontology-based mechanism to adapt to the third-party data source belonging to an arbitrary application domain.
4. Multiplatform portability.
5. High performance and high quality of rendering results.

Currently we are developing the new version of SciVi, which keeps using the advantages of ontology-driven paradigm and introduces new features facilitating the data visual analysis in a more comprehensive way. The features of the new version of SciVi are presented below.

3 From Visualization System to Comprehensive Data Analysis Toolset

While the visualization can help scientists to interpret the data obtained during experiments as well as to uncover hidden structures and regularities in these data, sometimes more elaborate analysis is needed than just rendering the picture. For example, if the data set is too big, it may be sometimes very useful to be able to filter out only a part of data matching given criteria. In other cases some new data should be generated according to the given set, for example, the trend lines, approximation lines or isosurfaces.

In these cases simple visualization tools should be improved to tackle visual analytics challenges. The main goal of visual analytics is to make the way of data processing more useful and transparent for an analytic discourse by combining automated analysis techniques with interactive visualizations. This contributes to effective understanding, reasoning and decision making, as pointed in (Keim D., Andrienko G., Fekete J.-D., Görg C., Kohlhammer J., & Melançon G., 2008).

To provide users with the extensible toolset for comprehensive data analysis we extended the SciVi architecture with data filtering subsystem and implemented it as a part of new version of SciVi. The SciVi 2.0 internal data flow is shown in the Figure 1.

As shown in Figure 1, there are three main stages of the new version of SciVi pipeline:

1. Adaptation to the third-party data source.
2. Filtering the data obtained from data source.
3. Rendering the filtered data.

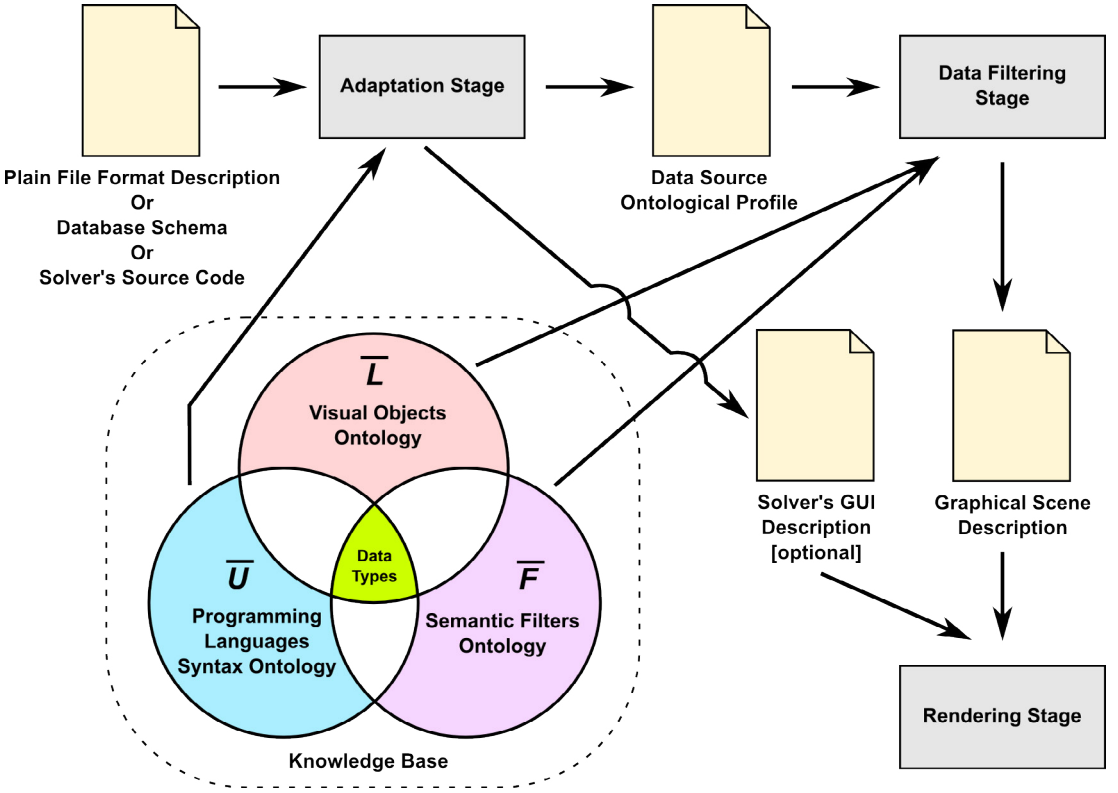


Figure 1: SciVi pipeline

These stages are discussed in the corresponding sections below.

4 Adaptation to Third-Party Data Sources

Currently, we propose several mechanisms to adapt to the following kinds of data sources:

1. Plain files.
2. Databases.
3. Hardware / software solvers.

To provide a uniform processing of these types of data sources and to meet their specifics we use ontology engineering methods. The adaptation subsystem (so-called integration module) creates the ontological profile for every data source according to the user's settings. This profile includes the data structure description and is used in the subsequent stages for adaptation.

4.1 Adaptation to Plain Files

In case of plain files, there is high-level graphical interface allowing the user to chose, whether the file is plain text or binary, to list the parameters stored in the file, to specify their types, order and delimiters, as well as to define the entries that should be ignored (for example, comment blocks, etc.).

According to the user's settings done in the GUI, SciVi generates related data structure description and data source file parser based on either regular expressions (for plain text files) or binary streams (for binary files). This parser is used to obtain the actual data from the data source file. Data structure

description is stored in a form of ontology (so-called data source ontological profile). Each element of this description has a reference number identifying either capture group of the regex-based parser, or the item number if the binary file parser. This reference number is used to establish a connection between the described elements and actual data in runtime.

Thereby SciVi can be easily adapted to arbitrary file format through the GUI, and additionally it provides an extensible array of presets for popular file formats (for example, PLY, 3DS, etc.).

4.2 Adaptation to Databases

To adapt SciVi to specifics of databases we propose to use the existing external tools, which are developed with participation of one of this paper authors. Papers (Chuprina S. & Nasraoui O., Using Ontology-based Adaptable Scientific Visualization and Cognitive Graphics Tools to Transform Traditional Information Systems Into Intelligent Systems, 2016) (Chuprina S., Postanogov I., & Nasraoui O., Ontology Based Data Access Methods to Teach Students to Transform Traditional Information Systems and Simplify Decision Making Process, 2016) describe how these tools are used to implement an ontology-driven web-service named Reply to provide a natural language interface to legacy information systems built on top of relational database management systems. The reported study was partially supported by the Government of Perm Krai, research project No.C 26/004.08 and by the Foundation of Assistance for Small Innovative Enterprises, Russia.

Because databases are structured data sources, the tools mentioned above are used to extract the database schema and to transform it into ontology automatically. Table names are mapped to ontology classes, foreign keys are mapped to object properties, and other column names are mapped to data properties. The transformation tools do not depend on the specificities of a concrete application domain and database management system.

OBDA (Ontology-Based Data Access) approach enables ontology-based accessing one or more data sources in a uniform way. The ontology has been generated from concrete database schema is interpreted within SciVi as a desired ontological profile of the data source.

4.3 Adaptation to Solvers

There are two kinds of solvers generating data to be visualized: hardware (different kinds of sensors, MRI scanners, DNA sequencers, etc.) and software (modeling programs, simulation systems, calculators, etc.). But usually in both cases solver has some software interface provided to retrieve the data. Typically this interface either provides API to get the data in runtime using some inter-process communication technique (like pipes, message queues, sockets, etc.) or stores the data in files or databases.

In case the solver produces files or stores its results in databases, the adaptation can be potentially done through the mechanisms described in sections 4.1 and 4.2. But in some cases there is a better way for the user described below.

If the solver's source code is available for the user (read-only is enough), the entire adaptation process can be automated. For this case there is ontology $\overline{L}$ (see Figure 1), which is a part of the SciVi knowledge base. This ontology describes syntax of input-output statements and variable definition statements of programming languages. It is used for automatically generating the parser that analyzes solver's source code and extracts the description of related input and output data structures. Thereby the stage of data structures specification is performed automatically. The only action the user has to do is to choose the data structure elements he/she is interested in.

Currently C, C++, Fortran-90 and Java are supported, but extending the ontology $\overline{L}$ to the new language is as easy as describing input/output statements of this language in Backus-Naur form.

In the current version of SciVi the communication with solver is performed via files produced by solver. The support of inter-process communication using various techniques is under development. This kind of communication enables in-situ visualization mode (Rivi M., Calori L., Muscianisi G., &

Slavnic V., 2012), which is extremely useful to cover the cases when the solver runs on high-performance computing system generating big amount of data. For these solvers it is not efficient to use any storage like files or databases, because the connection with hard drive device becomes a bottleneck, so inter-process communication is the only way to retrieve the data fast enough.

If the source code is not accessible to the user, it is still possible to adapt visualization system to the solver using mechanisms described in sections 4.1 and 4.2 with the help of high-level GUI to specify solver's input and output data structures manually.

As mentioned above, not only output data structures of the solver are described during the adaptation, but also input ones. This allows setting up the feedback from SciVi to the solver.

The description of GUI for the solver including control elements (buttons, sliders, text fields, etc.) is automatically generated based on the input data structures' specification and integrated into the SciVi client GUI. This enables user to control the solver from-within the visualization system, for example, to pause/restart the calculation or to change the input parameters.

The description of input and output data structure is stored in the form of ontology just like in previous cases, allowing the visualization system to handle solver like any other kind of data source.

5 Data Filtering and Uncertainties Highlighting

Once the ontological profile of the data source is composed, it is used in the next stage of the SciVi pipeline to set up the data preprocessing. This allows the user to change the data for visual analytical purposes without direct interfering in the data source and without changing solver's source code.

To enable data preprocessing we propose mechanism based on semantic filters. In general, the semantic filter is a mapping $\varphi: \langle I, S \rangle \to O$, where I is a set of typed inputs, S is a set of settings and O is a set of typed outputs. The set of filters is described by the ontology $\bar{F}$ (see Figure 1) that is a part of the SciVi knowledge base. For each filter this ontology includes the following descriptions:

1. Name.
2. Role (whether filter is operator or constant).
3. Set of inputs and outputs (filter's interface).
4. Set of properties (filter's settings).
5. Optional: source code of semantic filter's implementation in the case of nonstandard filtering algorithm (different interpreted languages like JavaScript, Lua or Python, runtime compiled shading language GLSL are supported) or alternatively link to the dynamically loading library.

The term "operator" is a common term to designate actions specified as regular expressions, compiled functions, and source code written in some programming language.

The fragment of the applied ontology $\bar{F}$ describing HSV to RGB color space conversion filter is shown in Figure 2. As shown, this filter is specified as a source code written in GLSL, which may be called as "hsv2rgb(%1)".

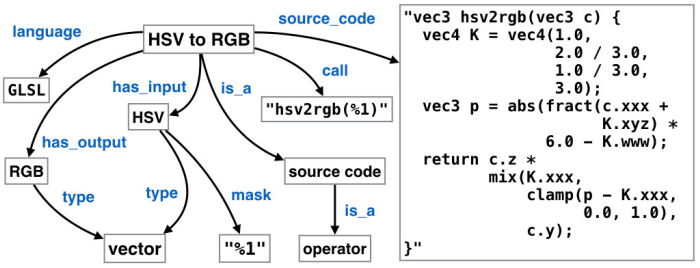


Figure 2: Ontological description of HSV to RGB color space conversion filter

The filtering stage is divided into two steps: primary filtering on the server-side and secondary (final) filtering on the client-side. User tunes primary filtering right after adaptation stage. It is assumed, that the primary filtering settings changes quite rarely (because they require user to connect to the server Web interface) and therefore serves for common data preprocessing purposes. Final filtering is set up before the actual visualization on the client-side. To improve the system's performance it is recommended to use this filtering to fine-tune the visualization system to the specifics of particular analytical task being solved.

Tuning of final filtering also includes setting up the output elements, which are the graphical objects depicted in graphic scenes. The properties of graphical objects and scenes available are described by the ontology $\overline{U}$ (see Figure 1). The fragment of this ontology describing the line-based wireframe object is shown in Figure 3.

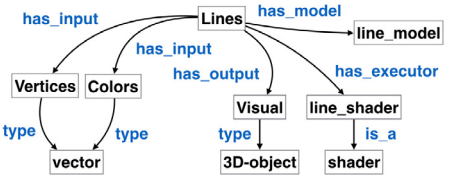


Figure 3: Ontological description of line-based wireframe object

Such kind of applied ontologies is used to control the pipeline's behavior and to represent type of the data passing through the data flow. According to the ontologies $\overline{F}$ and $\overline{U}$, the palette filters, graphical objects and scenes is automatically generated enabling the user to build data flow diagram. The comprehensive overview of dataflow principles is presented in (Lee B. & Hudson A.R., 1993). We implemented special node editor as a part of SciVi to help users to describe the concrete pipeline by means of data flow diagram within this editor.

The example of the data flow diagram created on the SciVi client-side for phylogenetic tree rendering is shown in Figure 4. The input data were obtained by bacteria strains DNA sequencing in Institute of Ecology and Genetics of Microorganisms (Perm, Russia) and processed by the third-party solver ClustalW (Larkin M., et al., 2007).

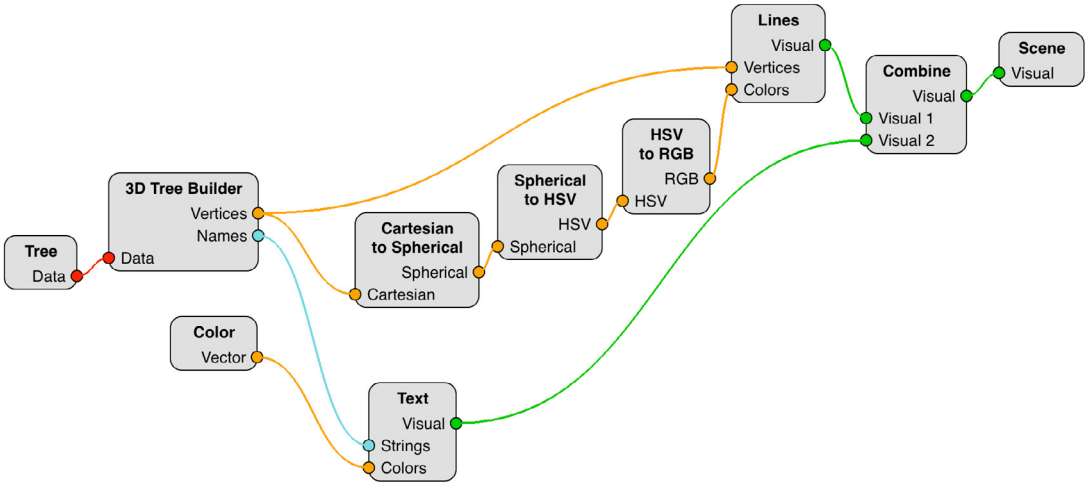


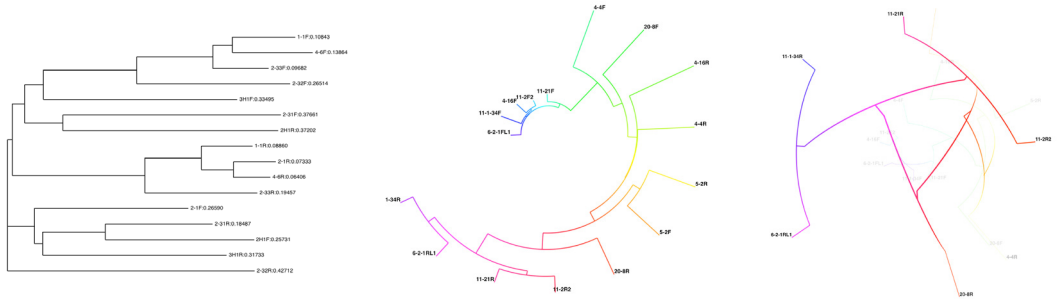
Figure 4: Data flow diagram for phylogenetic tree rendering on the SciVi client side

The node "Tree" represents data source ontological profile generated on the adaptation stage and obtained from the server. The nodes "Lines" (the ontology fragment describing this node is shown in Figure 3) and "Text" represent graphical objects to be displayed and the nodes "Combine" and

"Scene" represent graphical scene containing the objects; these four nodes are described in ontology $\overline{U}$. There is a set of standard data processing semantic filters in the SciVi ontology $\overline{F}$, which can be combined to construct more complex filters to process the data. In the given example, the nodes "3D Tree Builder", "Cartesian to Spherical", "Spherical to HSV", "HSV to RGB" (the ontology fragment describing this node is shown in Figure 2) and "Color" are used as filters to define data transformation from simple hierarchical representation of bacteria strains to another one enriched with additional information extracted from their relative positions within phylogenetic tree mapped to color values.

The settings done on the filtering stage described above are stored as a description of the graphical scene. This description has references to the entries of data source description from ontological profile and is treated as a scene template that will be filled with actual data on the rendering stage. The data flow paradigm is chosen as a well-established method to control data transformations. For example, data flow is used in 3D editors Blender and Maya, however those programs do not use ontology-driven approach to control the set of filters available. The distinctiveness of our approach is that the behavior of entire filtering subsystem is driven by the knowledge base and thereby can by modified and extended without changes of visualization system source code to adapt to the specific of research goals.

Different visual representations of the same data help scientists to analyze the results of experiments in more comprehensive way, to reveal the uncertainties in the objects being researched, etc. Compare, for example, the different rendering results of the same data. In Figure 5a the bacteria strains joined together in the tree are implied to have descended from a common ancestor. The semantic filters, which map bacteria genetic characteristics similarity to similar colors, help to increase phylogenetic diagram visibility, as shown in Figure 5b and Figure 5c. This may be one of the possible building blocks for insightful analytics of scientific experiments, which enable to turn comprehensive visibility into a model validation tool.

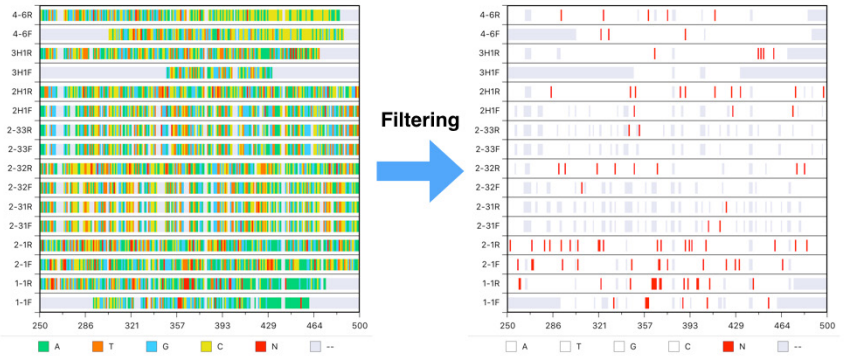


a b c

Figure 5: Simple hierarchy (a), 2D (b) and 3D (c) color enriched representations of phylogenetic tree rendered in SciVi

Adapting SciVi to solve real-world problems in different research areas we have found out that extending the visualization system functionality with adaptable ontology-based semantic filters allows analyzing input data uncertainties in different ways and thereby improving the comprehensiveness of visual analytic tools. For example, the Figure 6 shows the result of applying semantic filters to highlight the uncertainties in the bacteria strains DNA sequences. These uncertainties are the genome parts that failed to be recognized by DNA sequencer.

As a result, semantic filters can be used to validate data source, in particular, the solver. Thereby it enables to validate also the model the solver is based on. In the given example the highlighting of uncertainties help researcher to discover the quality of the source biological material as well as the quality of DNA sequencing soft- and hardware.



a b

Figure 6: The result of rendering the bacteria strains DNA sequences with no filtering (a) and after filtering that highlights uncertainties (b).

6 Rendering the Data

The rendering stage uses the graphical scene description to build actual visual objects and the solver's GUI description (if any) to integrate interface elements needed to steer the solver into the SciVi GUI. Rendering stage is described in details in (Ryabinin K. & Chuprina S., Development of Ontology-Based Multiplatform Adaptive Scientific Visualization System, 2015).

SciVi distinctive feature is adaptively distrusted rendering that ensures high visual quality and quick response of the visualization system to the user's commands. Three different rendering modes are supported: client-side rendering (ensures minimal response latency at the expense of high requirements to the client's hardware), distributed rendering (ensures optimal load in case of low client's rendering capabilities by performing reasonable data simplification and partial visualization on the server) and server-side rendering (ensures minimal requirements to the client at the expense of relatively high response latency). The appropriate rendering mode is chosen automatically according to client's performance, server load and network connection speed.

The examples of images rendered in SciVi are demonstrated in Figure 5 and Figure 6.

7 Conclusion

In this paper, we described the features of new version of our scientific visualization system SciVi 2.0 towards visual analytics. The distinctive features of our approach to develop high-level toolset for comprehensive visual analysis of complex data and model validation are adaptivity based on ontology engineering techniques and multiplatform portability.

The working process in SciVi is divided into three main stages: adaptation to the third-party data source (plain text or binary file, database or hardware/software solver), filtering the data obtained from the data source and rendering the filtered data. The advantage of the approach described is that developers will be able to adapt the system to their personal preferences and also to extend SciVi functionality without modifying the system's core code.

In the future, SciVi can be used to tackle the challenges of Big Data visualization:

1. Ontology-based data primary filtering performed on the SciVi server side with HPC facilities can reduce the Big Data *volume* either by sampling the only data pieces the user is interested

in, or by data aggregation (averaging samples, finding trend lines, etc.) during rendering taking into account the personal users preferences.
2. Ontology-based adaptation to different formats of data sources can tackle *variety* of Big Data.
3. Implicit parallelism of data processing in data flow paradigm can tackle *velocity* of Big Data: SciVi server can easily handle the primary filtering stage in parallel.

As described in (Ryabinin K. & Chuprina S., Using Scientific Visualization Tools to Bridge the Talent Gap, 2015), SciVi has been used to solve different visualization tasks from several application domains, such as physics, bioinformatics, medicine, etc. Now we already started developing the Big Data ready version of SciVi based on the principals mentioned above within a remit of the project titled "Socio-Cognitive Modeling of Social Networks Users Verbal and Non-Verbal Behavior Based on Machine Learning and Geoinformation Technologies". The reported study is supported by Ministry of Education and Science of the Russian Federation, State Assignment № 34.1505.2017/PCh (Research Project of Perm State University, 2017–2019).

Another way of SciVi improvement is building upon the semantic filters ontology the meta-ontology, describing the constraints of the filters' combinations. The problem is, that while the semantic filters ontology defines a set of filters available, there are two types of ambiguity: different combinations of filters can solve the same task and not every single ordered combination of these filters is meaningful. The meta-ontology can drive an automatic validation of the filters' sequences and help to discover the optimal filter chain.

References

Chuprina S., & Nasraoui O. (2016). Using Ontology-based Adaptable Scientific Visualization and Cognitive Graphics Tools to Transform Traditional Information Systems Into Intelligent Systems. *Scientific Visualization , 8* (1), 23–44.

Chuprina S., Postanogov I., & Nasraoui O. (2016). Ontology Based Data Access Methods to Teach Students to Transform Traditional Information Systems and Simplify Decision Making Process. *Procedia Computer Science , 80*, 1801–1811.

Keim D., Andrienko G., Fekete J.-D., Görg C., Kohlhammer J., & Melançon G. (2008). Visual Analytics: Definition, Process and Challenges. *Information Visualization - Human-Centered Issues and Perspectives* , 154–175.

Larkin M., Blackshields G., Brown N., Chenna R., McGettigan P., McWilliam H., et al. (2007). ClustalW and ClustalX version 2. *Bioinformatics , 23*, 2947–2948.

Lee B., & Hudson A.R. (1993). Issues in Dataflow Computing. *Advances in Computers , 37*, 285–333.

Rivi M., Calori L., Muscianisi G., & Slavnic V. (2012). *In situ visualization: State-of-the-art and some use cases.* Retrieved 01 30, 2017, from PRACE: http://www.prace-ri.eu/IMG/pdf/In-situ_Visualization_State-of-the-art_and_Some_Use_Cases-2.pdf

Ryabinin K., & Chuprina S. (2013). Adaptive Scientific Visualization System for Desktop Computers and Mobile Devices. *Procedia Computer Science , 18*, 722–731.

Ryabinin K., & Chuprina S. (2015). Development of Ontology-Based Multiplatform Adaptive Scientific Visualization System. *Journal of Computational Science , 10*, 370–381.

Ryabinin K., & Chuprina S. (2015). Using Scientific Visualization Tools to Bridge the Talent Gap. *Procedia Computer Science , 51*, 1734–1741.

Ryabinin K., Chuprina S., & Bortnikov A. (2016). Automated Tuning Of Scientific Visualization Systems To Varying Data Sources. *Scientific Visualization , 8* (4), 1–14.

Ryabinin K., Chuprina S., & Bortnikov A. (2016). New Ways to Adapt Scientific Visualization Systems to Third-Party Solvers. *Proceedings of GraphiCon2016* (pp. 126–130). Nizhny Novgorod: Nizhny Novgorod State University of Architecture and Civil Engineering.