ANATOMY OF A VISUALIZATION ON-DEMAND SERVER
A Service Oriented Architecture to Visually Explore Large Data Collections

Romain Vuillemot, Béatrice Rumpler, Jean-Marie Pinon
LIRIS, INSA Lyon, Université de Lyon, F-69621, Lyon, France
{*romain.vuillemot, beatrice.rumpler, jean-marie.pinon*}*@insa-lyon.fr*

Keywords: Visualization On-Demand (VizOD), Information Visualization (InfoVis), Service Oriented Architecture (SOA).

Abstract: Facing the relentless information volume increase, users are not only lost in information overload, but also among the various ways to depict it. In this paper, we tackle this issue by providing end-users access to up-to-date visualizations techniques, using remote services coupled to their local interactive environment. The outline of a Visualization On-Demand (VizOD) architecture is introduced, packaging information visualization processes into services reachable over a network. Our goal is to provide end-users flexible and personalized visual overviews of large datasets. We implemented a prototype partially validating our architecture, and discuss preliminary results of our experiments and give future work perspectives.

1 INTRODUCTION

In this paper we are interested in making visualization techniques broadly available to end-users according to his data, needs and task to achieve (see Figure 1). Availability is meant in terms of time-to-product, skills required and span of choice to offer users the right visualization technique, coupled into their local interactive environment (software and interactive devices), at the right time.

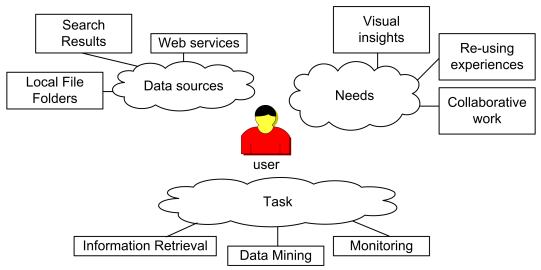

Figure 1: General user's data, needs and tasks.

Painted with broad strokes, the major *technical* issue with visualization techniques is the data format heterogeneity and the lack of reliability: there is a huge gap between a proof of concept issued from research works and a out of the shelve tool. Advanced contributions exist, but scattered in so many different application fields. For instance, visual data mining tools are very prolific in biology (Adai et al., 2004) with stunning results facing real life problems, especially dealing with data masses. New heuristics of finding patterns in huge structures are available for a specific tasks, but those inspiring data depiction remain domain specific. At the end of the day, end-users cannot benefit from most of the scientific tools even if they look inspiring and potentially useful. And finally, concerning interactive environments, the desktop metaphor is only still massively used because of universal availability: innovation cannot make its breakthrough.

As companies massively digitalize data for productivity, ubiquity and quality management, users would like to follow the behavior of massive and complex data evolution, coming from many sources. Whereas there exists tools to perform dedicated analysis of the data, as far as we know there is no way to get a visual overview carrying insights that will trigger a more complex investigation with a dedicated interface, regardless of the data or task (see Figure 2). In other words, there are no generic visualization setups dealing with both accurate and global information. Our goal is to uncover phenomena unseen at the

first sight, which will guide the user to a specific data subset or data projection.

Existing solutions are either basics visualization techniques such as sorted lists or unorganized items, never surpassing ordinary tables or spreadsheets. Thus, optimized access is only available when users change context by switching execution environment. But if no solutions are available then a whole product life cycle is started, inducing cost and time waste. This process also runs the risk of decreasing user's attention and productivity. New approaches have to be considered to keep users in the same interactive environment, with the same interactive devices he masters, just with a switch of data being analyzed.

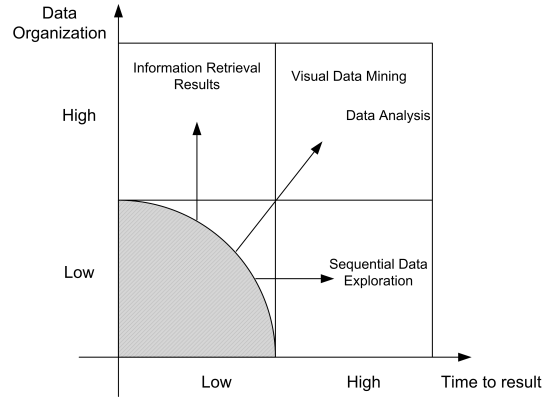

Figure 2: Generic overview (gray area) of data structures leads to more specific analysis.

Actors needs are identified as follow:

- **End-User Needs.** There is a need for abstract data handling solution to get an easy and quick visual insight of the data, with appealing visual metaphors. Such a process has to get rid of keyboard or not requiring any symbol being entered into the system (in case user cannot formalize his needs). Users have different background richness and cultures, so individual accessibility characteristic must be taken into account, such as visual deficiency.

- **Designer Needs.** They are experts in the field of translating user needs into software or assembly of software coupled with interactive devices. Whereas design are stored in a guideline format (Shneiderman and Plaisant, 2004), they lack formalization and evaluation. Re-use of existing libraries and environments becomes crucial with an increasing systems complexity and heterogeneity. The life cycle of products has to be etended as well.

- **Managers Needs.** Responsibilities involve advanced monitoring tools with high reliability. Trends are a way to anticipate the future and can raise by means of complex interactions analysis or long term data integration.

This paper is organized as follow. Section 2 focuses on the two major scientific fields being bridged, that are Information Visualization and Service Oriented Architecture. Section 3 focuses on the architecture outlines. Section 4 describes a prototype that has been developed. Section 5 discusses results and perspective. Section 6 concludes.

2 RELATED WORK

Our approach deals with Informations Visualization (InfoVis) techniques, which technical lacks are to be covered by web services packaging included in a Service Oriented Architecture (SOA). A synthetic comparison is given on Table 1, and similar attempts are also listed and analyzed.

2.1 Information Visualization

Conceptually, the fundamental goal of InfoVis is to find the right visualization at the right moment. A complete visualization process results in outputs such as maps to discover interrelationships between data. Relationships can be either internal or external, helping to understand complex static or dynamic dataset growing over time or actions. Human capabilities are thus enhanced, but limited cognitive memory must be taken into account in order not to overload users with information. The next step is for codes or user's knowledge to be integrated to reduce information dimension. Limited display space must also be considered, that can be solved by coordinated multiple views, either from a static perspective which consists of two or more distinct views supporting the investigation of a single conceptual entity (Michelle et al., 2000). Or either dynamically by synchronizing distinct views over time and/or user actions (Shneiderman, 1996).

Technically, the underlying problem is that visualization is hard coded to data and task to achieve. Today, reusing a technique for other data means starting another configuration/implementation cycle according to informal design guidelines. These recommendations lack formalization and are given in pattern format which has to be understood by experts, inducing extra costs. Another limit is that visualizations are local to user's application and dependent of his computing power. And because contributions are scattered in so many different application fields, there are neither central repository, nor evaluation. Some

lists exist, such as Many Eyes[1] or Visual Complexity [2]. The latter consists in an updated screenshot and video repository: but there are no semantic taxonomy providing automatic access to visualization.

2.1.1 Data Transformation Models

Our goal is to understand the visualization process in general, regardless data types, task or application domain. Then a model oriented analysis of the visualization has to be performed (Butler et al., 1993). This approach has been commonly used, resulting in many models. We focus on two major models from two distinct fields that are *Information* Visualization (Chi, 2000) and *Scientific* Visualization (Haber and McNabb, 1990). See Figure 3.

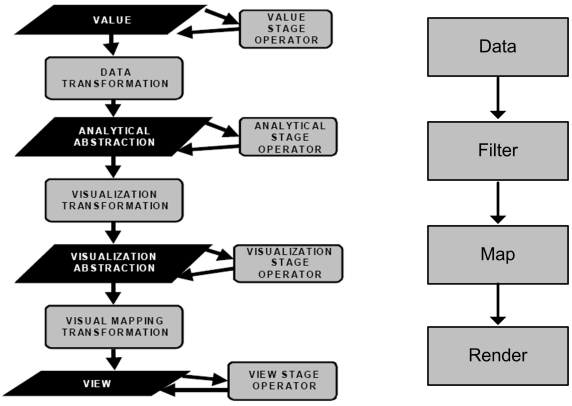

Figure 3: From left to right, (Chi, 2000) and (Haber and McNabb, 1990) models.

Both models consider the visualization process as a data flow, conceptually cut into steps: they end up with similar stages, but they are not complete enough as they do not integrate interactions. We proposed a slightly different model based on the previous ones, but including interactions: we don't consider the visualization process as a full data flow but merely as a sequence of operations being independently performed by means of actions. Our model decomposes the Information Visualization process into 3 stages, each coupled with user actions (see Figure 4). Interactions can be either manual (user's action), either semi-manual (user's action triggers a system reaction) or automatic (system's action).

The model description is as follow:

Extraction: is a step considering any perspective of visualization such as semantic (RDF), physical (matrix, directories) or even social, which extract different points of view from the data.

[1]http://services.alphaworks.ibm.com/manyeyes/

[2]http://www.visualcomplexity.com/

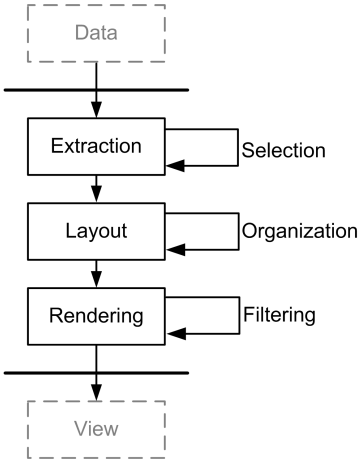

Figure 4: Holistic data transformation model including both data transformation processes and interactions.

Selection succeeds in reducing the dataset by selecting (SQL, SPARQL), aggregating and projecting in an appropriate manner, freeing the user from a potential overload.

Layout: is to give a spatial attribute to abstract data structure such as graph, lists or multidimensional sets. The layout can be made out of 2D maps but can also be a 3D model.

Organization is an action that will change the layout attribute of data.

Render: is to transform abstract data layout into images or 3D models.

Filtering means to *post*-process (using image analysis techniques such as Gaussian blur or Laplacian filter) to provide a *pre*-processed visual result (Vuillemot and Peralta, 2008). These mathematical computations based on 2D-signal transformations help, for instance, users to fade details or highlight contours.

The cut in layers helps to identify and describe each technique, that can now be seen as independent modular programs.

2.2 Service Oriented Architectures

Service Oriented Architecture (SOA) is an architecture style that aims at reorganizing business processes into loosely coupled packages. The packages are distributed into modules reachable over a network. The simplicity of use and universality of access makes it a design style helping the reuse of self-describing components. The components are interfaced as individual entities with the propensity to build applications very quickly. Services goal is to allow functionality to be stuck together to form ad-hoc applications reusing existing software service. The capabilities of adaptabil-

ity and evolution are high by adding a new service, reducing developing costs and a quick deployment.

Service are software reachable over a network independently from the underlying implementation. Languages describing them are:

- Interfaces are published in the WSDL file (XML-Based Document that describes how to communicate with the Web Service) (Christensen et al., 2001).
- Service repositories store WSDL files and are using UDDI (Universal Description, Discovery and Integration) helping to match with user's needs. (UDDI, 2000)
- Clients having retrieved relevant interface will contact the service provider using SOAP (SOAP, 2000).

Service composition (Agarwal et al., 2005) enables resources to be merged and responds more quickly and cost-effectively to changing market conditions. Service helps not having license/software distribution issues, and can reduce distribution costs, piracy and reverse engineering.

2.3 Visualization Service

The challenge of generically benefiting from visualization techniques has already been faced through many researches carried out in various disciplines. As far as we know, the closest and most prolific field is *Scientific* Visualization, providing related architectures. (Wood et al., 1996) introduced a client/server communication based on a simple reference model to perform data visualization. The visualization is done over the web, and focuses only on a specific data type which are plots. (Bonneau et al., 2005) introduces a modular visualization environments enabling users to change the data pipeline. A GUI interface is described, where dynamic change of the visualization pipeline can be done by users. Finally (Blazona, 2007) is a Scientific Visualization system offering web services facilities, but which is not based on a model. Thus visualization is seen as a whole process which steps can't be isolated.

Table 1: Characteristics of InfoVis and SOA.

Type/Field	**InfoVis**	**SOA**
Location	Local	Distributed
Coupling	Tight	Loose
Flexibility	Bundle	Package
Messages	File format	Messages
Communication	None	Protocol

3 A Visualization On-Demand Architecture

The key idea of the Visualization On-Demand (VizOD) architecture is 1) to separate into independent modules the visualization process according to our model (as seen on Figure 4) and 2) distribute processes, regardless the data being studied. This modular approach has to be combined according to a strategy, taking into account both technical constraints and users' preferences.

Using *on-demand* paradigm means we consider the rendered data stream as a media, such as movies with Video On-Demand (VoD) that can be chosen according to a user action. Conceptually, this expression is well-fitted since we consider the visualization as a stream of existing knowledge, just with another perspective. Technically, it holds many identical properties as VoD, such as cache, replication and performance.

3.1 Architecture

The aim is to make visualizations technique as black boxes, with a focus on interfaces and communication protocols.

3.1.1 Processes subdivision

Subdividing means making smaller parts (called business processes) of a larger system, operating independently. Each business process has properties (described in a UDDI repository) such as a description including the task it achieves, performance and complexity. These information will be useful to respect users Quality of Service constraints.

While communication among the processes becomes asynchronous, modules keep interdependence constraints. For instance a change in the layout will trigger a new render. A color change in the graph depiction will leave an identical layout, but here again render will have to be regenerated. We used as exchange protocol among the modules existing intermediate data transformation. For instance, the extraction module will communicate a graph structure to the layout process as it would be done in a monolithic architecture. In other words, external interface mirror internal temporary data residing in computer memory. The language choice turned out to be XML-like to wrap messages as showed on Figure 5. Then an additional SOAP communication protocol layer is added.

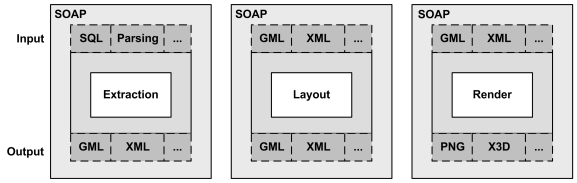

Figure 5: Modules are interfaced with specific languages, encapsulated in SOAP messages.

3.1.2 Processes Distribution

Business processes can be located at any places, and reachable through a service repository. Process distribution means that some processes may remain local to users (e.g. because of privacy issues) while other may be distributed and reachable other a network (e.g. because they require lots of computing resources).

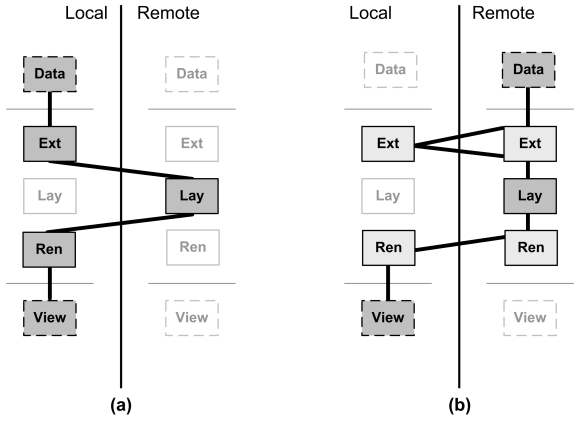

Figure 6: (a) strategy is to use remote layout process (b) strategy is to keep local interactions only.

There exists many distribution strategies that can be complex and dynamic (changing over time, service availability or service load). Two examples are described on Figure 6: (a) shows that user hosts the dataset and transfer only data structure to perform a layout process on it. This is a case where the layout needs lots of computing power (b) shows an opposite strategy, where user selects only the data and interacts with the render only. This is a case where a dataset is shared and reached through a local interactive environment.

3.2 Strategy of Access

A strategy is a common way of combining small steps to tackle a bigger problem. A step will be regarded as a group of processes, which are selected and glued together to solve a task. Assembly of steps will be done as close as the way the mind is working and will be called patterns. These patterns are following common orchestration that have been subject of study, such as Schneiderman's Overview Zoom and Details-On-Demand (OZD) (Shneiderman, 1996).

3.3 Personalization

While a company is an organization having the same focus, individuals have specificities to be considered. Even if every task or context may be different, there exists a visual knowledge about data structures (e.g. tree-like structures similar as Figure 8) that are common to every kind of information. Learning how to understand and master this knowledge requires time, but once done it can result in time savings by cutting delays to visually master new datasets. Thus, user's visual habits and preferences have to be identified and stored.

Personalization (Brusilovsky et al., 2007) is a way of taking user's preferences into account. Researches have been carried out in such direction as in information retrieval systems, in order to reduce datasets according to user's explicit or implicit preferences. *Explicit* preferences are user selections and configurations operations, such as local environment choice or service choice. *Implicit* preferences are user's history or any typical behavior registered in a non-intrusive way. For instance, if a specific service or group of services are invoked many times, they will be considered as a preference, even if no question has ever been asked to the user (Eirinaki and Vazirgiannis, 2003).

Preferences can also vary from short to long term interest. *Short term* preferences are edge colors, any encoded knowledge (such as symbols) and filters of the render which aims at solving a task in a specific context. *Middle term* preference is the data layout which can't vary (whereas colors can) in order not to disturb user's mental model, which is his internal vision of a virtual scene. Finally, a *long term* interest concerns the interactive environment in which are integrated visualizations.

Preferences will be stored in a *User Visual Profile* and will be available regardless user's location.

4 Prototype

To validate our architecture, we developed a first VizOD prototype using existing visualization techniques and interactive environments. Advanced technical details are available in (Vuillemot and Peralta, 2008). Our approach was to implement the strategy described in Figure 6 (b) which Use Case is available Figure 7.

The dataset is a movie database[3]. The data ex-

[3]http://www.imdb.com/

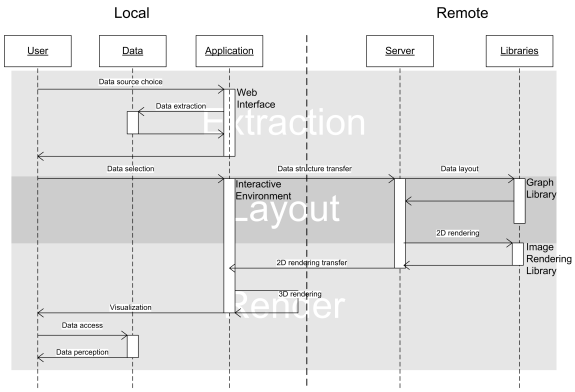

Figure 7: Prototype Use Case.

traction consists in performing queries, selecting 1) users and 2) for each users their rated movies. That selection is done by means of a web interface allowing SQL-like queries. The extracted result consists in a tree-like structure where the root is an artificial node connecting all users with their rated movies as leaves. The graph layout techniques used was originally aimed to display protein networks (Adai et al., 2004). Other graph visualization tools exist such as (Auber, 2003). The render step results in an image with annotated data (containing details about movies), provided in a separate file structured in a `XML`-like format, `KML` (Keyhole Markup Language). The result is the picture displayed on Figure 8.

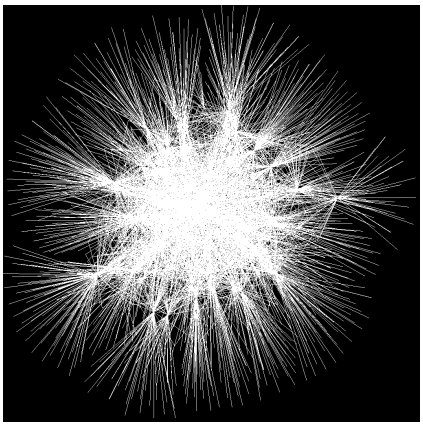

Figure 8: A tree-like layout visualization of a query result.

The image looks intriguing, but lacks of efficiency since it holds only structural information: it has to be included in a user's interactive environment and be more detailed.

We selected Google earth (GE) as an interactive environment (with all geo-spatial features disabled). A screenshot is available Figure 9. GE is installed and run by the client, and connects to VizOD by mean of HTTP requests. Using HTTP helps to keep away firewall issues or any complicated network configuration. VizOD will seamlessly map onto GE's 3D sphere an image according to user's altitude and angle of view, following (Shneiderman, 1996)'s recommendations. The result is a 3-layered multi-scale strategy combining external business processes, and resulting (by decreasing altitude) in an overview layer, zoom layer and details layer. The overview layer aims at showing global trends, then it will be connected to a blurring render service, to remove details and raise trends. The service interfaces Gimp[4] used in command line. The zoom layer remains the original image. The detail layer is the original image augmented with details on top of it (included in the `KML` file). Bandwidth usage has been minimized with that detail layer, by keeping the zoom image locally and only requesting and adding light `KML` data on top of it. The `KML` is converted to additional vectorial graphics (lines, captions, ..) by GE.

It takes about 59s (on an ordinary server) to generate a single image. The layout process is the greediest one, and generating or blurring images involves nearly no extra time or resource use cost.

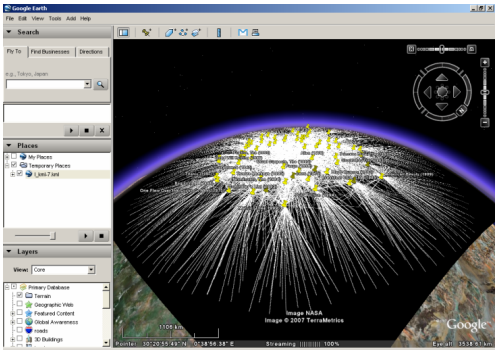

Figure 9: Image rendered mapped onto a 3D-sphere with details on top of it.

Results, issued from experiments, was that GE is a good metaphor since it implements a well-known object that is earth. And users already used GE for other purpose: then we managed to minimize the environment mastering phase by reusing an existing and widespread tool. Usability was excellent since we reused a powerful environment, which remains very reactive to every gestures and moves from users, even if images were not fully loaded or updated yet. Thanks to our VizOD approach, many visualization strategies can be adopted, while the client stays focused to a very same interface. The dataset can even totally change without absolutely no interface change. Adding new features on VizOD will be transparent for users. Finally users appreciated to use an attractive means that is a 3D sphere (similar to the *iPod*

[4]GNU Image Manipulation Program available at: http://www.gimp.org/

effect where the appealing wheel attracts users).

5 Discussions

In this section we discuss applications of our architecture.

Company benefits issued from using a VizOD architecture will come by outsourcing visualization to experts, providing software as services with better support and piracy prevention. The information system rationalization will go further by centralizing computing power at the same remote place and leaving end-users with lightweight heterogeneous terminals. The software maintenance routine is not on site any more but on the VizOD servers, holding business processes, which can be numerous allowing diversity and backups alternatives. New economical models will appear for producers.

Privacy and security issues are a big concern in our approach. Regarding the steps of our model one can see that rendered images prevent reverse engineering process. Furthermore, extracting data structures only will give structural information and preventing details being visible: quantity of information is given, not quality. Finally, a service approach prevents implementation details to be visible. However, we keep SOA related issues, such as messages exchange that is prone to attacks.

User's needs have to be considered globally, with imperfections, such as short attention spans. Memories are also important aspects to deal with, especially with data masses and in the case information is available in streams which are not stored: user's full attention is then required. The focus can be on visualizing updates, data growth, rather than content itself: the change or the behavior becomes as important as the instant content. The scalability and adaptability of the VizOD architecture is crucial.

Services Mashup interfaces are new way for users to compose services to build up and share new ones. But is does not include yet extra process such as data layout and render. A future work is to implement a *Visualization Mashup Interface* to cope with the lack of semantic and integration visualization service repository. User needs tools in new era where web-users have become actors using web interfaces.

New design process and product life cycle will emerge. Programmers are not constrained, then they will keep their own programming habits. A piecewise conception process can be set up, with progressive features. Other Research communities are addressed such as cognitivists in order to observe users behavior, interface designer to re-think the way to design and evaluate. New actors such as artists can now fully take part to the design process by including artistic among one or many navigation step.

6 Conclusion

In this paper we introduced the outlines of a visualization on-demand architecture (VizOD). We first proposed a holistic data transformation model, considering both visualizations and interactions. Every step of the model are considered as business processes that can either remain local or be distributed. Such an approach allows end-users to benefit and personalize visualization services, and couple it into their local interactive environments.

A present day result is a prototype resulting from an assembly of existing tools and showing that even non visualization-dedicated tools (such as graph manipulation libraries) can quickly result in an innovating application, following VizOD's specifications, in a context of affordable systems and non-expensive softwares.

The missing masterpiece in our approach is a way to encapsulate processes to make them automatically discoverable and usable. There is also a semantic gap between user's task needs which are expressed in natural language and tasks the machine already knows how to achieve. To tackle this issue, our next step is building on line communities that will fertilize best practices or novel uses. We will also focus on user generated data (e.g. traces of use) that have to be structured, filtered and sorted. More generally, a stable on line framework has to emerge and be sustainable over time, and then lessons will be learn by widespread use.

REFERENCES

Adai, A. T., Date, S. V., Wieland, S., and Marcotte, E. M. (2004). Lgl: creating a map of protein function with an algorithm for visualizing very large biological networks. *J Mol Biol*, 340(1):179–190.

Agarwal, V., Dasgupta, K., Karnik, N., Kumar, A., Kundu, A., Mittal, S., and Srivastava, B. (2005). A service creation environment based on end to end composition of web services. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 128–137, New York, NY, USA. ACM.

Auber, D. (2003). Tulip : A huge graph visualisation framework. In Mutzel, P. and Jünger, M., editors, *Graph Drawing Softwares*, Mathematics and Visualization, pages 105–126. Springer-Verlag.

Blazona, Bojan; Mihajlovic, Z. (25-28 June 2007). Visualization service based on web services. *Information Technology Interfaces, 2007. ITI 2007. 29th International Conference on*, pages 673–678.

Bonneau, G.-P., Ertl, T., and Nielson, G. M. (2005). *Scientific Visualization: The Visual Extraction of Knowledge from Data*. Mathematics+Visualization. Springer.

Brusilovsky, P., Kobsa, A., and Nejdl, W. E. (2007). *The Adaptive Web*.

Butler, D. M., Almond, J. C., Bergeron, R. D., Brodlie, K. W., and Haber, R. B. (1993). Visualization reference models. In *VIS '93: Proceedings of the 4th conference on Visualization '93*, pages 337–342.

Chi, E. H. (2000). A taxonomy of visualization techniques using the data state reference model. In *INFOVIS*, pages 69–76.

Christensen, E., Curbera, F., Meredith, G., and Weerawarana, S. (2001). Web Services Description Language (WSDL) 1.1. Technical report, W3C Note, http://www.w3.org/TR/wsdl.

Eirinaki, M. and Vazirgiannis, M. (2003). Web mining for web personalization. *ACM Trans. Inter. Tech.*, 3(1):1–27.

Haber, R. and McNabb, D. (1990). Visualization idioms: A conceptual model for scientific visualization systems. In *Visualization in Scientific Computing*, pages 74–93, G.M. Nielson, B. Shriver, and L.J. Rosenblum, eds, CS Press Los Alamitos, Calif. IEEE.

Michelle, Woodruff, A., and Kuchinsky, A. (2000). Guidelines for using multiple views in information visualization. In *Advanced Visual Interfaces*, pages 110–119.

Shneiderman, B. (1996). The eyes have it: A task by data type taxonomy for information visualizations. In *VL '96: Proceedings of the 1996 IEEE Symposium on Visual Languages*, page 336, Washington, DC, USA. IEEE Computer Society.

Shneiderman, B. and Plaisant, C. (2004). *Designing the User Interface : Strategies for Effective Human-Computer Interaction (4th Edition)*. Addison Wesley.

SOAP (2000). Simple object access protocol (soap 1.1). *http://www.w3.org/TR/SOAP*.

UDDI (2000). Universal description, discovery and integration, version 3. *OASIS, Billerica, Mass., 2000; www.uddi.org*.

Vuillemot, R. and Peralta, V. (2008). From Beautiful to Useful: A Multi-Scale Visualization of Users Movie Ratings. Technical Report RR-LIRIS-2008-001, LIRIS UMR 5205 CNRS/INSA Lyon.

Wood, J., Brodlie, K., and Wright, H. (1996). Visualization over the world wide web and its application to environmental data. In *VIS '96: Proceedings of the 7th conference on Visualization '96*, pages 81–ff., Los Alamitos, CA, USA. IEEE Computer Society Press.