Visual Analytics Infrastructures: From Data Management to Exploration

Jean-Daniel Fekete

Software and Hardware Infrastructures for Visual Analytics

Jean-Daniel Fekete, *Senior Member, IEEE*

Abstract—Visual Analytics is growing in popularity both as a research and application field. However, implementing Visual Analytics applications remains difficult due to the mismatch between the needs of analysts who perform exploratory analysis and existing software components for information visualization, data analysis, and data management. We explain analysts' requirements in terms of human cognitive abilities and show how information visualization has had to re-implement in-memory databases and some analytical algorithms to meet these demands. Visual Analytics relies on visualization but also requires tools for performing analysis and managing data at larger scales. We explain how the competing constraints of limited human cognition and increasing data strain existing software systems and provide some hints how to address them in the future.
We argue that addressing these requirements will benefit not only Visual Analytics tools, but also analytics and data management systems in general.

Index Terms—Visual Analytics, visualization, hardware infrastructures, software infrastructures.

1 Introduction

According to Thomas & Cook: "Visual analytics is the science of analytical reasoning facilitated by interactive visual interfaces." [1]. Visual Analytics relies on three essential layers: *visualization* to provide effective communication and exploration capabilities for humans to reason about the information extracted from the data, *analytics* to simplify and summarize the data to be exploitable by humans, to extract information from raw data, as well as to provide some guidance on its exploration, and *data management* to store, retrieve, and distribute the data efficiently to the other two layers.

Visual Analytics (VA) focuses on supporting exploration and interaction in the analysis of big and complex data. Big-Data management and analysis—already seen as key challenges—and are made even more difficult by the very nature of the exploration process. When facing new complex data, humans need overviews, summaries and the capability to drill down. This allows us to look for patterns and correlations, as well as discover emergent phenomena, empirical models, and theories related to the data.

For example, discovering fraud schemes in credit card transaction data involves searching for unexpected events in billions of daily transactions happening all over the planet. The amount of data is huge and new schemes appear frequently since thieves are actively and creatively working against the automatic detection mechanisms already built by banks. Therefore, bank analysts need to explore transaction data continuously trying to find anomalies and develop explanations for those anomalies (most of which are not actually caused by a fraud). To do this, bank analysts must focus on fraudulent transactions, understand their mechanisms, trace them back on the data, and take action to treat them.

To perform the exploration, the system needs to support the analysis by raising alerts on non-standard transactions. This kind of transaction mining has become standard now and is based on user profiling. It relies on an analytical processing applied to every transaction that should be sensitive enough to detect important problems but so sensitive that it creates many false positives that would overwhelm the analyst. Once a fraudulent transaction has been found, the analyst should not only cancel it but also understand the mechanism that made it possible. This requires finding similar frauds using a notion of similarity that is very specific to the scheme—for example, the place where it occured, the persons involved, the amounts or goods involved, etc.

This stage of analysis amounts to finding a similarity measure in the data, which involves multiple queries to the transaction database. One technical difficulty here is managing the latency of queries—when the analyst has a hypothesis in mind and queries the system, an answer should arrive in a few seconds; otherwise the analyst may forget the hypothesis and need to retrieve it when the results arrive. This requirement that results be fast, even at the expense of accuracy, is due to the unique limitations on human cognition—but that's all we have to make sense of the data for now. Currently, however, most database queries produce just the opposite; results that are entirely accurate but not bounded in time.

Continuing our scenario, the analyst will likely perform not only direct database queries but also more complex analytical queries involving machine-learning, and

• *J.-D. Fekete is with the AVIZ Project-Team of INRIA Saclay-Île-de-France, Orsay, France.*
E-mail: jean-daniel.fekete@inria.fr
Web: http://www.aviz.fr/~fekete

statistics. It may even involve more complex video processing if, for example, the fraud involves cash machines which take videos of the operator. Again, current analytics systems would perform these more complex analyses as accurately as possible, but in unbounded time. Once a series of analyses is started, the analyst would have to defer his exploration to a later stage to avoid waiting idly for an answer from the system. Instead, VA involves using faster, less accurate mechanisms to provide initial results quickly and improve the accuracy later if needed. For example, if the frauds are localized geographically to a few cash machines and the analyst wants to collect and classify the faces of possible accomplices, she can start processing videos recorded from these machines. During the processing, if she realizes that the fraudulent operator is always wearing a motorcycle helmet—easier to recognize on video than human faces, she will want to change dynamically the recognition algorithm. However, most analytical systems available are not meant for interaction. Instead, they run to completion with no or little feedback and provide no mechanism for interactive steering. When the analyst needs to change parameters, the system restarts from scratch, ignoring any information gathered so far—despite the fact that a large portion of analysts' queries are refinement of previous ones.

From the viewpoint of a software developer, implementing a VA application is difficult because the traditional software and hardware layers lack important services required by the human cognitive system to process complex information. In the next sections, we list these requirements, what has been done so far to address them and summarize the main issues to solve if we want to move beyond Big-Data management and support interactive exploration of Big-Data.

1.1 VA Layers

VA relies essentially on three layers for its software and hardware: data management, analytics, and visualization (Figure 1). All three have greatly improved in the last decade, albeit largely in isolation. Recently, however, data management and analysis tools have begun to converge using multiple technologies including grids, cloud computing, and general-purpose graphics processing units (GPGPU). High-performance computing on large amounts of data relies on the conjunction of new distributed data management technologies coupled with distributed computations.

Unfortunately, exploration has not been taken into account in these new infrastructures and it seems very difficult to introduce it without deep changes in both data management and analysis computation. In the next sections, we explain the main issues by first describing the information visualization reference model and extending it to VA, showing the mismatches between the current infrastructures and the needs of VA.

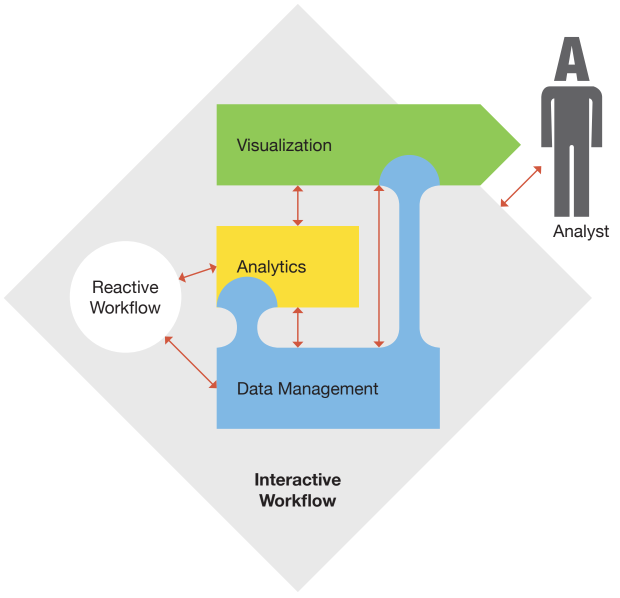

Fig. 1. The Three Layers of software and hardware and their relationships with higher-level workflows and the analyst. Red arrows represent control mechanisms whereas solid blue areas represent data sharing/distribution. The green area represents visualization as a communication medium between the system and the analyst.

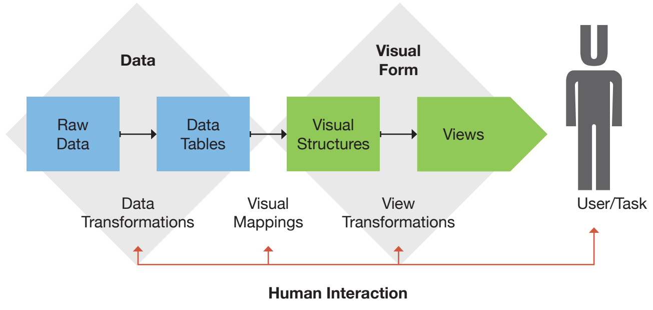

Fig. 2. The Information Visualization Reference Model according to [2]. On the top of the pipeline, data is adapted to be visualized and converted into an image through a visual mapping and a view. On the bottom, user interaction allows controlling all the stages of the pipeline with immediate feedback.

2 Visualization

Information visualization can be defined as the use of interactive visual representations of abstract data to amplify cognition [2]. Information visualization provides compact graphical presentations and user interfaces for interactively manipulating large numbers of items, possibly extracted from far larger datasets.

Information visualization applications all follow an abstract reference model that has evolved and matured, the latest iteration being described by [2] (Figure 2).

In the reference model, data flows from left to right.

Raw data is first transformed into proper data tables that are, in turn, transformed into visual structures through a visual mapping. The visual structures are then transformed into views (images) through a view transformation and presented to a user through a viewport. Interaction flows in the other direction. The user can control the view transformation parameters (essentially zooming and panning), the visual mapping parameters (*e.g.,* color mappings, layout for networks), as well as filtering from the database.

Most traditional information visualization toolkits model the visual structures as black-box components that manage their state internally and perform rendering (view mapping) directly to a window. This is called a *monolithic component model* [3]. In contrast, other information visualization toolkits model visual structures as regular tables with a specific schema where each data item has a geometry (or shape), an identifier, and a set of graphic attributes (*e.g.,* fill color, transparency, stroke color). This is called the *polylithic model* [3]. The arguments for monolithic vs. polylithic are similar to the arguments of object-oriented vs. database oriented structures. This debate is not yet settled, but it seems that it is easier to achieve scalability using a polylithic model because it can be implemented easily on top of standard databases. Monolithic approaches suffer from the so-called "impedance mismatch" problem, wherein it is difficult to convert their object-oriented structure into a database structure.

Tools that implement the information visualization reference model generally provide responsive interfaces where an interaction or change at any stage is propagated to the view very quickly. Ideally, interactive operations should happen in 100ms or less to appear instantaneous. Some operations are much slower, for example the computation of a large network layout. In that case, strategies are used to provide iterative solutions whenever possible to minimize idle waits for the user. Human-Computer Interaction knowledge on how humans work is now widely adopted in the domain of visualization. For example, it is well-understood that the reaction time of the system should be under 100ms and that actions should complete under 1-10s depending on the cognitive cost involved.

To comply with these requirements, information visualization systems typically rely on ad-hoc in-memory databases to hold the active data, and use a column-oriented architecture for performance reasons. It is interesting to note that, with a few exceptions, the existing implementations are not based on standard databases—the research fields of databases and visualization still need to join their efforts [4], [5]. This situation must change, since managing larger amounts of data will require faster and more sophisticated databases, which are too complex for non-specialists to implement. Conversely, existing databases will need extensions to manage the visualization data structures which are likely to become a standard functionality provided by database systems.

Fig. 3. VA on a Wall-Sized Display, link to http://www.youtube.com/watch?v=5i3xbitEVfs

In addition, recent visualization research has focused on improving analytical algorithms to match the reactivity requirement. For example, some graph layout algorithms are special kinds of multidimensional projections algorithms and much effort has been invested to implement these projections in an efficient and iterative fashion that shows the progress of the projection computation to the user. More sophisticated extensions allow users to steer iterative algorithms so that they devote more resources to computing analytical results that are relevant to the user's interests. These steerable algorithms are again variations of tradition analytical algorithms such as MDS, enhanced to match the user's cognitive process.

VA implies a scale change compared to information visualization, both in term of data size and in term of analyses required to make sense of the data. This scale change needs to produce software solutions that are fast and reliable, but which support the exploration process. In the next sections, we outline the state of the art in analytics and data management and explain why they do not yet meet the requirements of VA. We also offer some hints on how to improve them. These improvements would benefit to VA but also to the whole "Big-Data" community since they would provide more flexible mechanisms to handle Big-Data analysis in bounded time.

2.1 Hardware

One seemingly simple way of increasing the amount of visualized data is to increase the number of pixels. Indeed, several research groups and companies are exploring the venue of wall-sized displays (also known as "Power Walls") for visualization.

Extending the display to larger sizes requires developers to carefully consider screen size and resolution, the number of screens connected to each machine, how to distribute graphical applications to multiple machines, and how to handle input devices and interaction [6].

Despite these important initiatives, providing more pixels and better input devices will not solve the problem of visualizing large amounts of data since there are two important bottlenecks they do not address: perception and Big-Data. Human visual perception is limited by the physiology of the eye and the capacity of our brain to process images. Since the human eye is made of about 125 millions rods and 6 million cones as low-level receptors, we cannot perceive more features than that number—probably much less. Adding more pixels will allow more people to work in the same environment. However, it will not allow a single user to perceive more visual information. Moreover, many datasets available today exceed the capacities of even these displays by several orders of magnitude. For example, there is no way we will be able to directly visualize all the data stored by Facebook, not to mention Google. As a result, VA needs to resort to data analysis to summarize data and provide guidance on exploration.

3 Analytics

Analytics can be defined as the science or process of drawing conclusions from the analysis of data. Therefore, VA systems rely on data analysis and computer software that supports it. VA systems sometimes reuse existing data analysis systems, languages, and libraries, but also frequently re-implement them.

Systems and languages for data analysis have been used for a long time and are very diverse (the Wikipedia page "List of numerical analysis software" references 87 systems as of the writing of this article). These include general purpose languages such as "Matlab" or "R", as well as libraries for more specialized computations such as "OpenCV" for video analysis (http://opencv.org) or "GATE" for text analysis (http://gate.ac.uk/). All these systems follow the same simple workflow: load a data file, process it, compute some analytical quantities and export result files or statistical charts. Computation and analyses are often seen as black boxes that take tables as input and output, along with set of parameters, and run to completion or error without interruption. When managing reasonable amounts of data according to a streamlined process to obtain a final result, this is not a problem—the result will be obtained eventually and the analyst does not have to wait idly for it, even if the computation takes a long time to complete.

During exploration tasks, an analyst may need to try several processing methods to search for interesting results. If data is small, the analyses can be computed quickly and visualized using charting facilities in the analysis environment or by exporting the resulting tables to regular visualization systems for exploration. This pairing of visualization environments with analytical environments is used extensively in research and industry.

However, when data is large, such a loose coupling between visualization and analysis no longer works well for data exploration. The first issue here is that the data transfer time itself exceeds the reactivity requirement—moving data back and forth takes minutes, sometimes hours. To avoid costly data transfers, the developers of many analytical environments have added basic visualization functionality to their tools. However, the visualizations provided by most of these environments are less sophisticated than information visualization tools and are usually designed for publication and communication, not for exploration.

The second issue is that the algorithms provided by analytical environments are not designed for exploration and make no effort in providing early results quickly to the analyst. Again, this is a design issue that cannot be solved easily. Instead, it requires that processing and analysis algorithms be rewritten to take into account the human analyst in the loop. So far, the dominant strategy has been to optimize existing analytical environments that often rely on scripting languages. For example, "Riposte" [7] provides a just-in-time compiler for the R language that tries to use the multi-core capabilities of modern processors to speed-up standard vector operations in the R language (achieving speedups of 5-50 or more). While the same approach can be used to optimize Matlab or Python code, it merely pushes the limit of data sizes that can be effectively explored using analytical environments. This is very important, but does not directly solve the issue of scalability for VA.

The scalability issue of standard analytics has traditionally been addressed by high-performance computing—initially with expensive computers and, more recently, with affordable multi-machines architectures. These novel architectures include computer grids and cloud computing. However, in their current form, they offer high computation throughput along with high latency, making them ill-suited to the exploratory tasks of VA.

To manage large amounts of data, researchers and practitioners in VA currently need to re-implement existing algorithms themselves. Popular re-implemented algorithms include hierarchical clustering and principal component analysis computation, among other clusterings and projections.

Just like information visualization experts who have needed to re-implement in-memory databases, this is a sub-optimal use of skills and resources. Fortunately, as VA is becoming more visible, collaborations are underway to design and implement a new generation of analytical tools with better support for humans in the loop. This support will require anytime algorithms that trade accuracy for speed, and which are able to repair and continue computations instead of stopping and restarting from scratch.

4 Data Management

According to the Data Management International Association: "Data Resource Management is the development and execution of architectures, policies, practices and procedures that properly manage the full data lifecycle needs for an enterprise". VA applications need a reliable and efficient data management system layer to manage their data lifecycle.

Currently, SQL databases provide scalable storage, fast queries, and efficient distribution to load and save data through a network or shared-memory in a transparent manner. However, standard SQL databases are transactional—they assume that an application will select a small set of items, process them, and/or write a small set at a time. This assumption does not hold for VA, where, at any time, the analyst can request to visualize an unbounded portion of a database either directly or through an aggregate query. Therefore, most VA applications currently use one in-memory database and a different external data management system. Some visualization systems such as "Tableau" rely on a standard SQL database but still use their own in-memory database to hold visualization structures. This situation witnesses the mismatch between existing database technologies and needs for visualization, and we hope that, eventually, VA applications will be able to rely on more standard data management systems instead of building their own.

In recent years, there have been two trends to extend the capabilities of databases. NoSQL systems have relaxed some of the mandatory properties of standard databases—the so called "ACID" properties (Atomicity, Consistency, Isolation, Durability)—as well as the SQL language itself. Meanwhile, SQL systems have continued to extend their capabilities, adding high-performance features while keeping their ACID properties.

Some VA applications have even gone so far as to re-implement their data management layer in order to achieve good reaction times. For example, Fisher et al. have shown how aggregation queries can be implemented by returning running approximations [8]. Kersten et al. explain how fast incomplete queries could be implemented on MonetDB, a high-performance standard SQL database [9].

In the NoSQL world, Google has designed Dremel [10], a query engine designed to perform extremely fast queries on their infrastructure that "is capable of running aggregation queries over trillion-row tables in seconds." SAP has designed Hana-DB, an in-memory database system for its business-analytics applications [11]. This last example shows that VA is clearly of commercial interest but will require substantial investments in term of infrastructure to provide an acceptable user experience.

Overall, data management technologies are evolving quickly to provide better support for exploration and interaction. A few, such as Hana-DB, provide the important capabilities required to be usable directly for VA applications. However, the data management community has also begun to recognize the new needs raised by VA applications. In fact, some tools to address these issues, such as high-performance in-memory databases, are already usable. Still, more time is needed for VA developers and data management providers to develop infrastructure that fully supports analysts.

5 Integrated Environments

Although some work has been done to integrate the three software layers in a consistent framework, additional issues have appeared in VA applications that are not visible from the decomposition in layers.

5.1 Workflows for VA

Workflows are typically designed to carry-out complex and/or repetitive computations by sequencing computational and flow-control modules and running them in a database. Workflows are popular in business applications, such as banking, and are used to automate operations in a controlled way instead of relying on humans to perform risky or time-consuming sequences of operations. Scientific workflows that apply sequence of operations and flow-control to large corpuses of scientific data are also common. Because workflows are constructed in advance, multiple operations can often be performed in parallel, and most computations are designed to run from start to end with no interruption and little feedback.

Two new kinds of workflows have been designed specifically for VA: reactive workflows and interactive workflows.

5.2 Reactive Workflows for Dynamic Data

Traditional workflows run a sequence of processes each time database records are modified, while scientific workflows run a set of processes on a whole database once. For VA, however, some computations need to be performed on the whole dataset each time something in it is changed. This is important when managing dynamic data, since operations can be performed ahead of time to prepare it for interactive exploration. For example, a company interested in monitoring the reputation of its products can perform sentiment analysis for all web pages that mention them. This analysis can be performed continuously whenever new pages are fetched rather than at analysis time. The EdiFlow system has been designed to do just this and allows analysts to specify a set of operations that should be performed on the data each time it changes [12].

A naive approach would run the whole workflow for the whole database each time it is updated. This approach would be very resource-consuming and impractical when the changes occur faster than the time it takes to finish the computation. EdiFlow introduces several optimization strategies to avoid recomputing everything from scratch—instead, it tries to "repair" what has already been computed. The actual strategy for repairing is decided by each computation module according to the semantics of its computation. In the same vein, processes can decide to interrupt or repair their long computations when they are notified that the data table they operate on has changed.

Reactive workflows require non-standard support from their underlying database. First, they need a notification mechanism to control when the workflow needs to be restarted. The standard mechanism for notification in databases relies on “triggers”, but substantial work is needed to manage asynchronous notifications and communicate with an external workflow system each time a database modification occurs. To allow several processes to compute on the same tables without conflicts, the notion of transactions needs to be extended. In the EdiFlow prototype, every record of our tables is explicitly time-stamped to support long transactions. In parallel, Oracle has also implemented this feature in an extension of their database called “Total Recall” [13]. This illustrates how the requirements of VA are often already being considered by industrial actors.

5.3 Interactive Workflows with Provenance

Traditional workflows are created once and run several times. For VA, exploration itself can be viewed as the construction of a workflow. Here, a workflow is a cascade of operations that filter, summarize and analyze the data and ask: What are the right operations to apply? What parameters should be used for these operations? What is the difference between applying one sequence of operations versus another sequence? All these questions arise when exploring data to understand it or to turn it into a usable form. At intermediate stages, visualization can show the results of data processing, and can provide additional feedback and quality control measures.

VisTrails has been designed to interactively build and run scientific workflows that include visualization [14]. It is now a mature platform but continues to evolve quickly due to the very large number of application domains where it is applied, including physics, biology, medicine, climate-data modeling, and more. It is implemented in Python and can run native applications or use libraries once they are wrapped into VisTrails Python modules. Writing these wrapper modules in easy and a large number of packages has been integrated into VisTrails. This part is relatively standard for workflow systems in general. The novel part of VisTrails is its facilities for keeping track of changes in the workflow. These changes are recorded using a “provenance management” subsystem, allowing analysts to iteratively improve their workflow while keeping track of the changes. This provenance facility is not limited to one person. Instead, analysts can share workflows for interactive exploration with others in order to help them understand a process, re-apply it, or continue it.

However, the architecture of VisTrails still relies on applications and processes exchanging data through the standard input/output operations. Chaining analysis and visualization of large data can take a long time because of the cost of data communication. Currently, VisTrails does not enforce any data management mechanism, which allows it to remain agnostic about it, but does not allow it to optimize the data communication part of the workflow.

5.4 Abstraction Layers for VA

To overcome the problem of sending data through files or networks to communicate between analysis and visualization modules, the *Obvious* meta-toolkits provides an abstraction layer [15]. The cost of data transfer is responsible both for adding delays between modules, as well as for wasting memory by duplicating data between all the modules that need it. Obvious is implemented as a Java library and is designed to encapsulate other Java libraries. It relies on the fact that the all the classes used to implement data tables in software libraries are very similar—if not identical—except for variation in method and class names. This is true for visualization systems as well as for analysis systems and database systems. Therefore, Obvious provides an abstract class definition that can be used as a pivot to share data for all the encapsulated Java libraries. To wrap a library, only two classes are needed—one maps a native data table to an Obvious data table and the other maps an Obvious data table into a native data table. These mappings mostly involve calling methods with other names—they do not introduce noticeable slowdowns in accessing data.

Once a library is wrapped, it can exchange its data with other libraries almost for free. Therefore, database modules can produce Obvious tables that can be processed by machine-learning modules “in-place” and passed to visualization modules without additional input/output or duplication of data.

The compatibility layer between all the data-table models is not the smallest common set of functions provided by all the libraries. Rather, it is a superset that also specifies a semantic of notification that is usually not consistent between libraries. Currently, Obvious wrappers are defined for 4 visualization toolkits, 2 machine-learning toolkits, and the standard Java API for database access. It shows that the data communication issue can be addressed by designing a data service that can be efficiently mapped from multiple libraries. This library-neutral in-memory data abstraction is a very important data management component for VA.

6 Challenges

Collaboration between the fields of VA, visualization, analytics and data management has already started and many of the issues discussed in this article are increasingly being recognized by commercial vendors. It is interesting to note that special database infrastructures suitable for VA have been implemented by companies such as SAP, Google, and Oracle before they became popular in research, proving that VA is already an identified market for these companies.

At the data management layer, in-memory databases with support for extended data types (*e.g.*, geometric shapes) are needed by VA and are starting to be available with some vendors, although the extra services are not standardized yet. Once the required services are available, visualization systems should adopt them instead of relying of their own limited implementations. More research needs to be conducted in databases to offer mechanisms for trading accuracy for speed in queries, supporting query refinement, and adding new types of long transactions for expensive computations.

Using data abstractions on top of database services, as demonstrated in Obvious, it might be possible to extend data sharing to additional languages such as Python or R. Doing this could avoid data transfer altogether in VA applications and workflows such as VisTrails or EdiFlow.

This table-sharing/caching service would be different from the transaction-based service offered by most databases, allowing to implement much more efficient optimizations from inside the database driver than on top of transactions, as explained in [9].

At the analytics layer, more research needs to focus on designing analysis modules that can repair computations when data changes, provide continuous feedback during the computation, and be steered by user interaction when possible. At a higher level, balancing the tradeoff between speed and accuracy may call for new analysis strategies, as well as tools that can change between strategies depending on the analyst's current goal. More research needs to be conducted to provide these high-level analysis strategies to analysts.

At the system level, integration of visualization, analytics, and data management will require more collaboration between researchers and practitioners across all three fields. Just as information visualization relies on a well understood and accepted reference model, we need a reference model for VA. However, more time and experience from all domains is needed to design it.

7 Conclusion

VA relies on three important layers: visualization, analytics, and data management. Existing software and hardware architectures in these three domains are not suited to integration into VA applications yet, mostly because the need for exploration in large data has only been identified recently. In particular, the constraints imposed by human cognition are not well known outside of the information visualization field, but have strong implications on the services required for effective exploration of large amounts of data.

This article has outlined the current limitation of the three layers, but also shown some prototypes and solutions to address these limitations. Some solutions are mature enough to spread to production software, while other libraries will require more research or experiment to mature. Eventually, VA could become easier to implement and more widespread when the main issues listed in this article are solved.

In addition, providing the mechanisms for exploration in databases and analysis systems will benefit to all the situations when users are willing to trade accuracy for time, an important issue since time is becoming one of our most important resources.

Acknowledgments

Thanks to Wesley Willett for his help in improving the paper and Jeremy Boy for the illustrations.

References

[1] J. Thomas and K. Cook, Eds., *Illuminating the Path: Research and Development Agenda for Visual Analytics.* IEEE Press, 2005.

[2] S. Card, J. Mackinlay, and B. Shneiderman, *Readings in information visualization: using vision to think.* Morgan Kaufmann, 1999.

[3] B. B. Bederson, J. Grosjean, and J. Meyer, "Toolkit design for interactive structured graphics," *IEEE Trans. Softw. Eng.*, vol. 30, no. 8, pp. 535–546, Aug. 2004.

[4] J.-D. Fekete and C. Silva, "Managing Data for Visual Analytics: Opportunities and Challenges," *IEEE Data Eng. Bull.*, vol. 35, no. 3, pp. 27–36, Sep. 2012.

[5] Jean-Daniel Fekete, "Infrastructure," in *Mastering The Information Age - Solving Problems with Visual Analytics*, Daniel Keim and Jrn Kohlhammer and Geoffrey Ellis and Florian Mansmann, Ed. Eurographics Assoc., 2010, ch. 6, pp. 87–108.

[6] M. Beaudouin-Lafon, S. Huot, M. Nancel, W. Mackay, E. Pietriga, R. Primet, J. Wagner, O. Chapuis, C. Pillias, J. R. Eagan, T. Gjerlufsen, and C. Klokmose, "Multisurface interaction in the wild room," *Computer*, vol. 45, no. 4, pp. 48 –56, april 2012.

[7] J. Talbot, Z. DeVito, and P. Hanrahan, "Riposte: a trace-driven compiler and parallel vm for vector code in r," in *Proceedings of the 21st international conference on Parallel architectures and compilation techniques (PACT '12).* ACM, 2012, pp. 43–52.

[8] D. Fisher, I. Popov, S. Drucker, and M. Schraefel, "Trust me, i'm partially right: incremental visualization lets analysts explore large datasets faster," in *CHI '12*, 2012, pp. 1673–1682.

[9] P. A. Boncz, M. L. Kersten, and S. Manegold, "Breaking the memory wall in monetdb," *Commun. ACM*, vol. 51, no. 12, pp. 77–85, Dec. 2008.

[10] S. Melnik, A. Gubarev, J. J. Long, G. Romer, S. Shivakumar, M. Tolton, and T. Vassilakis, "Dremel: Interactive analysis of web-scale datasets," in *Proc. of the 36th Int'l Conf on Very Large Data Bases*, 2010, pp. 330–339.

[11] F. Färber, N. May, W. Lehner, P. Große, I. Müller, H. Rauhe, and J. Dees, "The sap hana database – an architecture overview," *IEEE Data Eng. Bull.*, vol. 35, no. 1, pp. 28–33, 2012.

[12] V. Benzaken, J.-D. Fekete, P.-L. Hémery, W. Khemiri, and I. Manolescu, "EdiFlow: data-intensive interactive workflows for visual analytics," in *ICDE 2011*, 2011, pp. 780–791.

[13] Oracle Inc., "Total recall white paper," http://www.oracle.com/technetwork/database/focus-areas/storage/total-recall-whitepaper-171749.pdf, Sep. 2012.

[14] S. P. Callahan, J. Freire, E. Santos, C. E. Scheidegger, C. T. Silva, and H. T. Vo, "VisTrails: visualization meets data management," in *SIGMOD Conference*, 2006, pp. 745–747.

[15] J.-D. Fekete, P.-L. Hémery, T. Baudel, and J. Wood, "Obvious: A meta-toolkit to encapsulate information visualization toolkits — one toolkit to bind them all," in *VAST 2011*, 2011, pp. 89–98.

Jean-Daniel Fekete received his PhD in Computer Science in 1996 from Universitée Paris-Sud. He was recruited by INRIA in 2002 as a confirmed researcher and became Senior Research Scientist in 2006. He is the Scientific Leader of the INRIA Project Team AVIZ that he founded in 2007.

His main Research areas are Visual Analytics, Information Visualization and Human Computer Interaction.