6 Infrastructure

6.1 Motivation

Supporting the strong demand in data storage, computation and interactive performances required by visual analytics applications is still a challenge. All currently existing visual analytics applications need to build their own specialised infrastructure for their specific problem. This is a normal stage of evolution in a domain of information science, as explained by Brian Gaines in his BRETAM model[46] (Figure 6.1). This model suggests that all research domains and fields run through the same stages. A new research domain or phenomenon starts by a Breakthrough – a discovery that can be physical or conceptual – followed by a Replication stage when the scientific community tries to replicate the initial discovery and test its limits. The next stage is Empiricism when researchers find empirical laws that can be used to apply the discovery. After that, some Theory is found that allows a deeper understanding and usually makes predictions about the phenomenon. The next stage is Automation when the phenomenon is totally accepted, followed by the Maturity stage when it is used routinely without question.

Visual analytics is only at the Replication stage

As the model describes, each domain should pass several stages before reaching maturity and this chapter plots a possible path to achieve this evolution successfully and effectively, when visual analytics as a whole is only at the Replication stage.

Visual analytics is both user-driven and data-driven

One of the most difficult issues of visual analytics is that it is both user-driven *and* data-driven. It is user-driven because during the interactive steps of the analysis, the user is specifying algorithms and parameters to explore the data. It is also data-driven because new data is made available to the user at unpredictable times, such as when algorithms run or databases are updated,. Traditionally, the domains described in chapters 3, 4 and 5 have created software infrastructures that are mostly data-driven with some exceptions for geographical visualisation systems. Conversely, visualisation systems are user-driven and manage mostly static data.

Software not designed for interaction or dynamic data is very difficult to adapt

Therefore, assembling technologies created by these multiple domains is a difficult challenge because the software infrastructures they currently rely on are incompatible at a deep level: when software is not designed to react to changes in the data or triggered by the user, it is very difficult to modify it later.

Interactive systems used to drive the analysis need to provide sub-second reactions to the user's actions. Furthermore, visualisation systems, required to understand large datasets visually, require the screen to be updated in less than 100ms following user action. In contrast, current databases serve transactions in

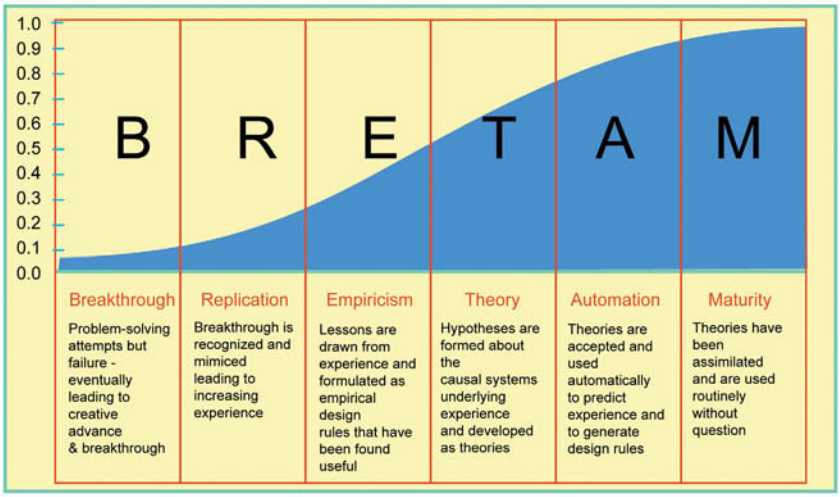

Figure 6.1: The BRETAM sequence plotted along the underlying logistic learning curve[46]

seconds and data mining algorithms available today are meant to run until completion, which can take minutes, hours or even days.

To overcome this situation, practitioners in visual analytics have started to implement ad-hoc systems, such as in-memory databases or user-steerable algorithms. However, this is not a sustainable solution in the long term for several reasons:

- **Loss in quality** When visualisation practitioners need to implement visual analytics systems, they cannot use off-the-shelf data-storage components or data mining components and hence need to implement them with their often limited skills of the domain. If data mining practitioners need to implement the same system, they will have problems integrating visualisation and interaction to a dynamically running analysis system. The problem is similar for a data-management practitioner.
- **Loss in resources** Since there is no accepted software architecture reference model for visual analytics, each system implements its software components in slightly different ways, leading to incompatibilities and no interoperability. This is becoming a bottleneck in the evolution of the field because most of the modules needed are difficult and expensive to implement and the lack of interoperability hinders sharing them within the community.
- **Loss in research time** Because research groups have to re-implement the visual analytics modules they require, they loose valuable time that would be better used for innovation.

Standards in visual analytics components will create a new market

- **Lack of component market** Since no standard exists, no commercial market can emerge for components. Several European companies sell visual analytics components but their market remains small at this level compared to other software components.

6.1.1 Examples

By taking the role of various actors in visual analytics, the software infrastructure issues are much easier to understand.

Exploration of Hierarchical Clustering from an Information Visualisation Viewpoint

Hierarchical clustering is one of the most popular clustering techniques used to make sense of large datasets. A large number of items (e.g., documents, genes, files, persons) are grouped according to a similarity criterion. Documents can be grouped according to the similarity of their textual contents, or simply because they share an author. Genes can be grouped because their DNA sequences are very similar, etc. The outcome of hierarchical clustering is a tree (or a direct acyclic graph) and the information visualisation community has a long tradition, as well as a collection of visual representations and interaction techniques, to navigate such trees. So, once the data has been hierarchical clustered, it can be visualised and explored using well-known and effective techniques.

However, in real life, computing good and meaningful hierarchical clustering is difficult and a push-button approach to clustering is likely to produce an incomprehensible hierarchy. Several issues should be considered when performing such clustering: what similarity measure to use, what attributes to select, how to deal with outliers and missing values, to name a few. The statistical analysis community has extensively studied these issues and also provide a wealth of quality measures to validate clustering, but choosing the similarity measures, the attributes and the validation method add extra complexity to the process that is now essentially made by trial and error.

Very few systems have effectively combined information visualisation with hierarchical clustering. HCE[96] is one example specialised for biological applications. It has required its author, a specialist in information visualisation, to re-implement popular hierarchical clustering algorithms and similarity metrics computation to offer the level of interaction required to achieve successful clustering. However this work is only applicable to one applied domain and therefore cannot be used in other domains. Breaking down such an application in modular components that could be assembled to suit other application domains in a modular, extensible and reusable way is currently not possible, due to the lack of a unified architectural model and appropriate software architecture to support it. Furthermore, to meet the interactive demands, the algorithm itself has to be programmed by an information visualisation specialist. Apart from the loss of time for the specialist, it may limit the level of sophistication of analytic components added to the information visualisation application since few information visualisation specialist are also specialists in statistical analysis.

Mining and Exploring Long Time Series from a Data Mining Viewpoint

VizTree[71] is a visual tool for mining large time series (Figure 4.5). Instead of using time dependent values directly, VizTree encodes it as a complete tree with a width and depth that can be specified interactively. The data mining part is very smart since the change from a long series of value into a tree simplifies greatly, many kinds of computation and allows for interactive search of patterns. However, the graphical interface of VizTree is very simple and the interaction is limited, with simple interactions such as selection of a time-range being done through form-based entries rather than by direct manipulation. Furthermore, VizTree is meant to mine long time-series, but as it reads flat files rather than make use of a database, its range is restricted. Again, the authors were specialised in one domain and did not use a ready-made software framework to implement their visualisation and interaction in a more effective way; they had to re-implement the missing parts in the best way they could, far from the state of the art in information visualisation, HCI and data management.

Database and Other Viewpoints

Good engineering practices imply separation of concerns without sacrificing quality

Further examples of this kind can be seen in the database field or from other kinds of analytical domains (video analysis, text analysis, etc.). The message here is that to build real applications, all these domains need each other's expertise, but currently, due to deep infrastructure model incompatibilities, they cannot connect the pieces of software together. Once all these domains agree on a conceptual model and implement it in their tools and modules, interoperability will become possible and cross fertilisation will become simpler.

6.1.2 Conclusion

To build demanding visual analytics applications, we need a new conceptual software architecture, a good separation of purpose between different stages of this software architecture and a good decomposition in components. Once we have agreed on this architectural model, we can create a new market of high-quality interoperable components to build the applications needed to transform the current flood of data into an opportunity for discoveries. These components, commercial or free, would allow researchers to focus on their domain of interest and skills and to push the research forward effectively. They will also increase the competitiveness of commercial companies by allowing them a better understanding of the market trends.

Designing the conceptual architecture is not simple because it is both user-driven and data-driven. Visual analytics is based on empowering human analysts and allowing them to apply complex analytical computations while maintaining interactive feedback and control. Most current analytical components are designed to run without interruption, delivering their results at the end.

For large datasets, this can take hours or days. Visual analytics needs analytical techniques that adapt to the human analysis process, not the other way around. As quoted by Thomas[111], chapter 5:

> Create visual analytics data structures, intermediate representations and outputs that support seamless integration of tools so that data requests and acquisition, visual analysis, note-taking, presentation composition, and dissemination all take place within a cohesive environment that supports around-the-clock operation and provides robust privacy and security control.

Even when these components are well understood, even standardised, more research on data typing is needed. Current databases only expose storage types (e.g., bytes, long integers, etc.) but not their meaning. Exposing the semantic type of a data is essential, in order to know what kind of analyse can be applied and what kind of visualisation is meaningful. An integer value can be used to represent a nominal identifier such as a hash value, it can also represent a day of the week or month or a true numeric value. SQL databases do not express the semantic of the numbers stored. Data mining systems usually classify data as nominal, ordered, numerical and ratio. This classification is rich enough for most statistical treatments but not sufficient for visualisation. The semantic web is an example of an application domain where sophisticated data types are being defined but there are also other initiatives and it is not clear how they will converge.

Visual analytics needs more expressive data types than provided by SQL or statistics

Since the requirements of visual analytics involve deep changes of the architectural models and implementations of several computer-science domains (e.g., databases, statistics, machine-learning, data analysis, visualisation), there is a strong need for all these domains to be aware of these demands to support exploratory and analytical capabilities in future systems.

6.2 State of the Art

Architectural models exist for all the domains related to visual analytics. We will briefly describe them and highlight the issues encountered when trying to incorporate them in visual analytics applications.

6.2.1 Visualisation Architecture and Data Structures

The domains of scientific visualisation and information visualisation have designed two reference architectural models that are slightly different but are now adopted in all the existing systems. The historic Visualisation Pipeline (Figure 6.2), proposed by Haber & McNabb[52] mainly describes the mapping of data space into visual space whereas the newer Information Visualisation Reference Model (Figure 6.3) as described by Card, Mackinlay and Shneiderman[25], which is a refinement of the Data State Model described by Ed Chi[29], refines the pipeline into a loop where user interaction can happen at all stages of the pipeline. All the well-known implementations of

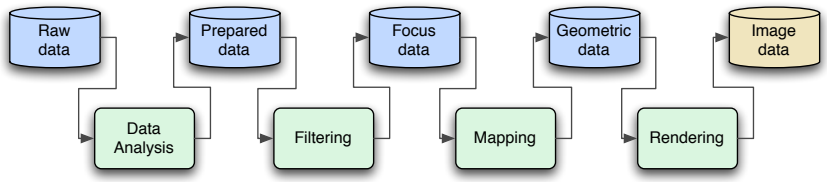

Figure 6.2: The Visualisation Pipeline, adapted from Dos Saltos and Brodlie[37]

information visualisation systems and toolkits adhere to this model and are mostly compatible conceptually, albeit slight implementations variations that give rise to some incompatibility problems, but efforts are ongoing to solve the interoperability issues.

While this model is useful for understanding the transformation from data to views and the processing of interactions back to the data, it fails to describe the analytical process of visual analytics.

Furthermore, the visualisation pipeline emphasises geometric data much more than information visualisation because much of its technical issues come from representing and optimising the geometry for rendering, which is of lesser concern to information visualisation.

Geographical visualisation reference model emphasises multi-scale representations at the data level and layering at the rendering level

Geographical visualisation is similar to scientific visualisation in the sense that geometry plays a very important role and that several methods have been used to model and encode the geography as geometric objects. Furthermore, most geographical visualisation systems are mostly 2D, so the final rendering stage is simple in principle but complex in practice due to the use of layers of information in most GIS systems. One important issue of geographical visualisation is the management of aggregation since maps show different levels of details with different forms depending on the zoom level. This issue of dynamic aggregation and multi-resolution modelling appears also in scientific visualisation but mainly for rendering issues. The problem of aggregation and multiple representations is much newer in information visualisation and has not been modelled in the existing architecture reference model. This is clearly

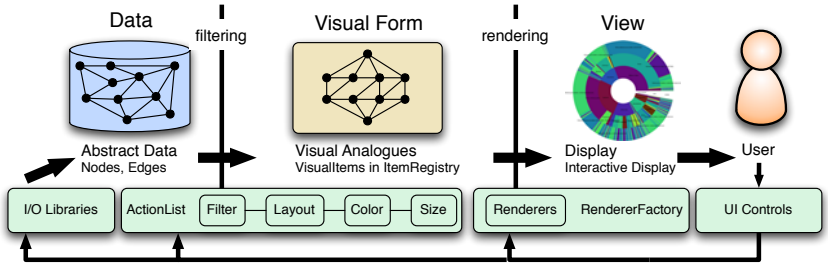

Figure 6.3: The Information Visualisation Reference Model, adapted from Heer et al.[57]

a visual analytics issue that should be better tackled by all the visualisation communities.

Blending different kinds of visualisations is currently difficult

Blending different kinds of visualisations in the same application is becoming more frequent. Scientific visualisation and geographic visualisation need information visualisation because they manage multi-valued data with complex topologies that can be visualised using their canonical geometry. In addition, they can also be explored with more abstract visual representations to avoid geometric artefacts. For example, census data can be visualised as a coloured map but also as a multi-dimensional dataset where the longitude and latitude are two attributes among others. Clustering this data by some similarity measure will then reveal places that can be far away in space but behave similarly in term of other attributes (e.g., level of education, level of income, size of houses etc.), similarity that would not be visible on a map.

On top of these visualisation systems, a user interface allows control of the overall application. User interfaces are well understood but they can be very different in styles. 3D systems use specific types of interfaces that are very different to traditional desktop interfaces. Moreover, information visualisation systems tend to deeply embed the interaction with the visualisation, offering special kinds of controls either directly inside the visualisations (e.g., range sliders on the axes of parallel coordinates) or around it but with special kinds of widgets (e.g., range sliders for performing range-queries). Interoperability can thus be described at several levels. At the data management level, at the architecture model level and at the interface level.

6.2.2 Data Management

All visual analytics applications start with data that can be either statically collected or dynamically produced. Depending on the nature of the data, visual analytics applications have used various ways of managing their storage. In order of sophistication, they are:

- Flat files using ad-hoc formats,
- Structured file formats such as XML,
- Specialised NoSQL systems, including Cloud Storage,
- Standard or extended transactional databases (SQL),
- Workflow or dataflow systems integrating storage, distribution and data processing.

Data Management for visual analytics can rely on different levels of sophistication

We will now consider these data storage methods, paying particular attention to the levels of service required by visual analytics, such as:

- Persistence (they all provide it by definition),
- Typing,
- Distribution,
- Atomic transactions,
- Notification,
- Interactive performance,
- Computation.

Ad-hoc flat files

Flat files, including XML, will only remain a commodity for interchange and high-performance acquisition of data

In the last 20 years, the most popular system for data analysis has been the spreadsheet calculator. Spreadsheets are ubiquitous and one of their strength is their simplicity and versatility, which comes partly from their lack of enforced strong typing and structuring. Most business and scientific data is available as spreadsheet files that are quite difficult to analyse automatically, due to this lack of typing and structuring. Therefore, practically all data analysis and visualisation systems provide extensive import/export support for spreadsheet files.

Variants of spreadsheet format files, such as the simple Comma Separated Values (CSV) files, are supported by almost all data-exchanging programs nowadays. The main pitfall of these spreadsheet formats is its lack of typing and metadata. These files require human interpretation before they can be used by an application.

Besides these well-known file formats, most data-oriented applications have used ad-hoc formats to save and load their internal state. The XML format has been designed to solve that problem and to offer an all-purpose file format for all the applications that require data storage and interchange. There are still three reasons not to adhere to standards: legacy programs that continue to use their ad-hoc formats, secrecy to hide details of internal structures of a system, and performance. XML adds sophisticated typing to flat files, which is very important, but no other services.

Highly demanding systems use ad-hoc architectures to collect data and analyse them quickly. Formats like the Hierarchical Data Format (HDF[1]), designed with performance in mind, are required in special situations, such as managing data returned from high-resolution high-throughput sensors in physics experiments, producing megabytes of data per second. However, this data is usually collected in short bursts and can be processed afterward using more standard formats. This step of data cleaning and filtering is an integral part of visual analytics and therefore, some visual analytics applications should be able to process and understand these formats, as well as the more standard ones. High-performance storage systems offer the same level of service as flat-files.

Traditional Databases (Row-based)

Extensions to traditional databases needed for typing, in-memory caching and fast notifications

Transactional databases have a long tradition of success and reliability. SQL is the standard and several products are currently available that implement different levels of SQL functionality for various prices, from free to thousands of Euros or more.

SQL technology is mature and implementations are usually robust – based on tables stored in row order. SQL provides atomic transactions (the well-known ACID properties). They provide most of the services required by visual

[1] `http://www.hdfgroup.org/`

analytics applications, except that the typing is not as expressive as needed. SQL types are related to their storage and to some extent to the operations that can be performed on them, but important properties of data cannot be expressed in a portable way using SQL alone. For example, standard SQL use integers for values and for categorical data (e.g., zip codes). It is essential in visual analytics (and statistics) to know precisely, the semantics of attributes in order to apply meaningful computations and visualisation techniques to them.

Since transactional databases implement all the data management services required for visual analytics, it would make sense for visual analytics systems to rely directly on them. However, they have several pitfalls:

- Interactively visualising data requires data to be in memory. With the exception of in-memory databases, standard transactional databases do not guarantee the sustained performance required by visualisation and analytical computations. Therefore, visual analytics components have to implement an in-memory version of the databases.
- The data types provided by SQL are mainly storage oriented, not semantic oriented. A value representing a latitude or longitude will be typed as Real. Visual analytics applications need to add more metadata and there is no widely adopted standard to do that.
- Notification is implemented through *triggers* in standard transactional databases. The trigger mechanism is very inefficient in most database implementation; some databases provide workarounds but they are not standard. Without an efficient notification mechanism implemented from the database layer, the visual analytics application needs to implement one on its own.

Analytical Databases (Column-based)

To address efficiency issues, both in terms of speed and memory, new databases architectures are column-based. For example, Kdb+ can handle streaming data and analysis on the fly; it has been experimented with in visual analytics by Chan et al. at Stanford. MonetDB[18] is a more general-purpose transactional database engine developed at CWI in Amsterdam that is also column-based. It implements most of the services required by visual analytics but has never been used as the default storage engine for visual analytics application so it remains to be seen if MonetDB delivers what it promises.

Specialised NoSQL Systems

Trendy NoSQL systems are spreading but their heterogeneity and short life-span are problematic

NoSQL systems are usually built to avoid the complexity of general transactional databases and provide faster, simpler or more specialised services. Google internally uses a very optimised file system called BigTable. Amazon internally uses a proprietary key-value structured storage system called Dynamo for its Web services. Several very different services are provided by NoSQL system, from document stores (e.g., CouchDB) to graphs (e.g., Neo4j), key-value store (e.g., BigTable) and hybrids.

NoSQL systems also include storage solutions on the Web or in 'Cloud Storage'. There is a trend in migrating resources on the Web through more than one provider. For example, several large online service providers (e.g., Amazon Simple Storage Service, Google Storage) provide Cloud Storage to allow outsourced storage and computations from Web services. Along the same line, new repositories on the Web offer high-level Web services to query their contents (e.g., Google and its visualisation API Data Source). However, ad-hoc storage management solutions do not provide any time performance guarantees for access or modification, so visual analytics applications need to build layers, such as caching, on top to deliver acceptable response.

Workflow and Dataflow Systems

According to the Workflow Management Coalition[2]:

> Workflow is concerned with the automation of procedures where documents, information or tasks are passed between participants according to a defined set of rules to achieve, or contribute to, an overall business goal. Whilst workflow may be manually organised, in practice most workflow is normally organised within the context of an IT system to provide computerised support for the procedural automation and it is to this area that the work of the Coalition is directed.

Scientific workflows have a great potential to become the backbone of visual analytic applications

In the recent years, several workflow systems have been designed to automate scientific processes; they are called 'scientific workflows' and since 2007 have their own workshop (IEEE International Workshop on Scientific Workflows)[3]. Although workflows are designed to apply a well-known process repeatedly, exploratory workflow systems are starting to appear, such as VisTrails [4]. VisTrails is system managing provenance and parameter setting for visualisation systems. A pipeline of processes is built and run interactively. Its results, in the form of visualisations, can be displayed in a table format, which allows multi-dimensional exploration by changing parameter values. The changes are recorded in a database, so later on, the user can explore their own construction of the pipeline or send it to another user for their interpretation. VisTrails is a very compelling system for exploration and visualisation of large-scale data. However, VisTrails has some weaknesses for visual analytics:

- It relies deeply on the Visualisation Toolkit (VTK): the visualisation pipeline is built directly as a VTK pipeline and parallel computation and rendering relies on the ParaView extension of VTK. Therefore, it relies heavily on a specific technology.
- It does not use a standard database for storing its state and data. It uses XML files in a directory to keep track of its history. VTK is neutral in term of data sources and can read from a large number of file formats and databases.

[2] http://www.wfmc.org/
[3] http://www.extreme.indiana.edu/swf-survey/
[4] http://www.vistrails.org

- It does not manage dynamic data: changing data sources does not trigger any re-computation and also each user initiated re-computation must start from scratch. VisTrails maintain a cache of computed results but the cache mechanism is not aware of dynamic data.
- It does not implement any protocol to manage the interaction among workflow/dataflow components. Only standard interactions are available.

Despite these weaknesses, VisTrails is a sophisticated scientific workflow system that allows exploration and provenance management. It should certainly be an inspiration for the future of visual analytics software infrastructures.

VisTrails should be an inspiration for future visual analytics software architectures

Data Management Conclusion

Ideally, the native storage management layer of a visual analytics application should provide all the services described in this section. Unfortunately, no existing storage management system currently offers all the required set of services. The visualisation community has started to design its own set of data management facilities that will not scale whereas the data management community is not yet aware of the new requirement for interaction and visualisation.

6.2.3 Data Analysis

Analytical systems usually implement a very simple architectural model: they read inputs and write outputs until their work is done. Several environments are available for analysis depending on the data types:

Analytical systems usually implement a very simple architectural model

- Statistical analysis (e.g., SPSS, SAS, R)
- Scientific computation (e.g., Matlab, Scilab)
- Machine learning toolkits (e.g., WEKA)
- Textual analysis (e.g., GATE, UIMA, SPSS/Text, SAS Text Miner)
- Video analysis (e.g., OpenCV)
- Image analysis (e.g., Khoros, IRIS Explorer)

Data analysis takes data as input (from a data management layer) and processes it to produce different kinds of interpretation. Analysis environments take several forms:

- Program libraries,
- Components,
- Toolkits,
- Simple applications,
- Integrated applications,
- Web-services.

Most of the analysis systems can be run from a database or flat files. They tend to be neutral in the form of the data they input, except for integrated applications

that internally manage some form of database. All of the analysis systems output their results in flat files, XML files or databases.

Data analysis and data management solutions are usually well integrated using open standards

Data analysis and data management solutions are usually well integrated and applications, that are performing analysis without user exploration, can be developed with a wealth of software solutions that can be combined, relatively easily in powerful ways; this combination is purely data driven: the program will run to completion to return a solution.

However, from a visual analytics standpoint, their main weakness lies in their 'architectural reference model'. Analysis systems read from data files, apply a computation and output to other data files. This is fine as long as interactivity and exploration are not required. For visual analytics, interactivity and exploration are essential. In the case of a large dataset and complex analysis, the analyst does not want to wait minutes or hours if they are not sure that the analysis is useful. In the past, systems such as Hive[92] have been designed for trading quality and speed, but they were only prototypes. To provide the right level of service, analytical components should change their reference model to be able to present an overview first and then allow for progressive refinement under the control of the analyst. More research is needed to better understand how to address these needs.

Analytical components should change their reference model to suit the needs of visual analytics

In the recent years, there have been several attempts at providing machine learning and data analysis as external services. For example, Microsoft has defined a set of data analysis protocols: XML for Analysis, DMX (Data Mining Models) and the Data Mining Group [5] has designed PMML (Predictive Model Markup Language) as a way to communicate and run data mining algorithms in a vendor neutral fashion.

These services are now available from several robust analytical platforms, either free such as the R statistical system or commercial such as SPSS, or toolkits such as Weka (a popular data mining toolkit in Java[6]). Web-based implementations are also available from well-known providers such as Google with its Google Prediction API[7]. From a visual analytics standpoint, these system offer a very rich set of capabilities but with crucial features missing:

- **Fast initial response with progressive refinement** Some analytical algorithms are incremental by nature. For example, computing eigenvectors with the largest eigen values (e.g., for Principal Component Analysis) uses a power-iteration that is incremental in nature; best heuristics for computing the 'travelling salesman algorithm' start from an initial tour and try to improve it incrementally. These algorithms and many others are routinely available in several analysis systems but they never provide any intermediate results due to the absence of a software protocol to send results on demand. A standardised protocol would be one way of overcoming the problem.

[5] http://www.dmg.org/
[6] http://www.cs.waikato.ac.nz/ml/weka/
[7] http://code.google.com/apis/predict/

- **Re-computation following small changes** Computing a hierarchical clustering is usually done in two steps: computing a distance matrix and iteratively extracting the two closest items until all items are extracted. The first operation has a quadratic complexity in the number of items to cluster, which can be very large. Once a distance matrix is computed, changing one item only means re-computing one line of the distance matrix, which is very fast. However, most clustering algorithms will not be able to store a distance matrix once the clustering has been computed in order to do this. Therefore, the penalty of changing a value (adding or removing) is quadratic instead of linear. Therefore, visual analytics systems need specialised dynamic hierarchical clustering functions (e.g., to remove outliers) rather than rely on standard library routines.
- **Steering the computation** Some algorithms are inherently slow, such as MDS (Multi-Dimensional Scaling). However, they can be steered to deliver faster results on a specified region of interest. This steering is not inherently complex to implement, but visual analytics practitioners need to do it themselves.

All these capabilities, and maybe some more, are required by visual analytics applications and are not provided by analysis systems. More research work is needed to find the right level of services and combination of algorithms to fit the needs of visual analytics.

Conclusion for Data Analysis

Services essential for visual analytics are missing from standard analytical architectures

In recent years, the different analysis communities have made a strong effort to facilitate the use of their systems and algorithms. Most advanced analytical systems can be connected to other applications through several mechanisms of communication: direct library linking, component integration, inter-process communication or Web services. However, there is a deep mismatch between the level of services they provide and the needs for visual analytics: they do not provide mechanisms for:

- Fast imprecise answers with progressive refinement;
- Incremental re-computation, either in the data (e.g., some data has been changed) or in the analysis parameters;
- Steering the computation towards data regions that are of higher interest to the user.

These newer requirements can be difficult or impossible to implement on top of the existing services. More research work is needed to understand how they can be supported in the future.

6.2.4 Dissemination and Communication

Currently, results of complex analysis are presented to decision makers using slide-shows such as Microsoft PowerPoint. A slide-show is a support for story-telling: analysts need to collect evidence of their findings to report them in

Few systems to manage dissemination and publication of visual analytics results

a meaningful order using text, still or animated images captured from their exploration and sources.

In the recent years, the GapMinder system designed by Hans Rosling [8] showed how effective visualisation and animation can be for telling compelling stories about data. However, existing visual analytics systems provide no mechanism to move from the analytical process to the presentation process, except for producing still images. Even these still images are not completely adequate for paper publication because the graphical characteristics of the printed medium are different from the screen.

Several systems have been designed to gather analysis information during an analytical process. The accumulated information can be revisited and kept for later use or archived. Oculus nSpace is one of them, designed for visual analytics applications. Although designed to help the exploration process, nSpace is also helpful to create presentations at the end of the analytical process. However, again, the created presentation is not interactive or linked to the actual exploration. Systems such as VisTrails (see Section 6.2.2) offer all the capabilities for linking back images to the exploration process. Explorations done with VisTrails can be distributed and replayed easily. However, they are currently not meant to be used for slide-shows style presentations.

To summarise, systems to manage dissemination and publication of visual analytics results are definitely lacking; this offers interesting opportunities for research and commercial products.

6.2.5 Cross-cutting Issues

Each domain has been exploring cross-cutting issues separately, they should now coordinate

Software infrastructure has been described above in the order of the pipeline process. However, issues that are common to all levels are now discussed.

Distribution

Distribution is an important aspect of visual analytics. The data management can be distributed, the analysis can be distributed and the rendering can be distributed. Therefore, several questions arise: is there one mechanism for distribution (for example, the database engine should be responsible for the distribution) or should there be one mechanism for each tool, or a general mechanism (for example multicast communication) so that all the tools can communicate using a common bus?

For now, each tool implements its distribution mechanism and visual analytics applications need to cope with all of them. Accessing a distributed resource, whether for storage, computation or rendering, is not particularly complicated and making use of several mechanisms is not an important issue, unless rapid interaction and coordination is involved. In that case, notification mechanisms should be used, which is complicated when several resources are involved

[8] http://www.gapminder.org/

because the mechanisms offered can be quite different. Standard SQL databases offer only triggers that are usually inefficient and no standard mechanism is provided to propagate notifications across a network. Analytical modules often do not provide any notification mechanism, with the exception of image processing systems, which usually do.

Even if one distribution mechanism could be used for all the parts of a visual analytics application, it might be less effective than several mechanisms well designed for each task. For example, the parallel visualisation rendering system ParaView[9] uses distributed memory, whereas most SQL databases use Internet network connections. There is no way to change either of these implementations for technical and historical reasons.

Finally, with the advent of computation in the Cloud, processing will also migrate to the Internet or to large computation grids. These technologies require special skills that are currently in short supply.

New Computing Paradigms

Beyond distribution, new programming paradigms are emerging. Cloud computing has already been mentioned, with its grid-computing variant, but GPU programming is also becoming more widespread and can be used for very demanding applications. Visualisation has been using GPU programming from early on, but the data analysis community is only starting to utilise this powerful computation resource.

All these new paradigms will evolve quickly in the forthcoming years and it is necessary for the visual analytics software infrastructure to keep pace with these developments and be compatible with them.

Language

Since visual analytics relies on several complex components to carry out potentially long computations, the programming language and interoperability between languages is very important. Currently, the choice of programming language used in a visual analytics project seriously restricts the choice of tools available. The information visualisation community has several toolkits programmed in Java. The scientific visualisation community generally uses C++. New environments such as Microsoft .NET allow programs written in different programming languages to interoperate but the Java language is not so well supported. New languages are now in use such as Microsoft F# for advanced functional programming, Scala for scalable computation and SVG-based JavaScript for Web application. New ones will eventually appear. How can visual analytics avoid constraining the software infrastructure landscape by programming languages? Two choices are possible:

Only research can teach us what combination of languages and mechanisms are best suited to develop and deploy visual analytics applications

[9] http://www.paraview.org/

- Rely on a virtual machine such as Microsoft CLR or the Java virtual machine, but there are still complex issues in term of code libraries that are not solved by a virtual machine.
- Use Web-based mechanisms such as Web services. However, whilst the current mechanisms can provide relatively high throughput it is usually at the expense of high latency, and therefore not suitable for interactive applications.

Only research can teach us what combination of languages and mechanisms are best suited to develop and deploy visual analytics applications beyond the current level of craftsmanship.

6.3 Challenges

Designing an accepted conceptual architectural model is difficult because it involves several well established domains

Designing an accepted conceptual architectural model for visual analytics is a difficult issue because it involves several domains that are already well established and hence will need a collaborative effort to understand cross-domain issues. Several workshops have started to tackle the problem but it can still take several years before reaching a consensus. More effort should be devoted to experiments in this domain so as to quickly agree on a recognised architectural model that all components comply with.

So far, visual analytics systems have been implemented by extending existing environments. Database practitioners have extended their database environment, machine-learning and data analysis practitioners have extended their analysis environments, and visualisation practitioners have extended their visualisation environments. The results are not satisfactory. This has led to work being done by non-experts in the fields, often leading to sub-optimal solutions; too many resources have been wasted to 'reinvent the wheel' and the solutions do not scale or do not provide good quality interaction.

A unified architectural model does not mean one unified implementation

A unified architectural model will involve fairly new programming paradigms such as asynchronous computing and the management of multi-scale data structures. It is important to emphasise that a unified architectural model does not mean one unified implementation. Several domains have found it necessary to deal with this issue in the past and have found several solutions without relying on one particular implementation. However, in contrast to previous standardisation work, visual analytics will involve much more diverse domains and some clear methodology should be devised to reach convergence and agreements among this diversity.

Once this conceptual phase is achieved, it will lead to a clear specification of software components and to the potential creation of a market for components. Practitioners of visual analytics applications will be able to reuse components implemented by others, whether commercial or free, whether for profit or for research. Designing analytical components that scale and provide capabilities for interaction is a difficult challenge. It will require new analysis methods, in addition to the adaptation of existing methods that have not been implemented with interaction in mind.

Moreover, the requirements of visual analytics will foster new interesting research in the domain of high-performance databases, analytical components that can provide results at any time and be steered interactively, and new visualisations that could scale to arbitrarily large sized datasets.

6.3.1 Visualisation

Visualisation architectures should merge; more research is needed to solve incompatibilities

Existing visualisation software infrastructures are quite different in capabilities. Scientific visualisation can manage terabytes of geometric data in real-time with special-purpose computers, as information visualisation can only deal with millions of data points. Geographical visualisation can only display a limited amount of information, usually less than a million items, but by using very sophisticated aggregation methods that can manage terabytes of actual data, users are able to navigate freely. The important challenges are thus:

- Allow all the visualisation domains to share a common rendering pipeline, where graphic acceleration can be used simply, multi-thread rendering is supported natively, and overlaying and other merging techniques can be used to blend images generated from all the visualisation domains (scientific, information-based or geographical).
- Improve research on data structures and algorithms for aggregation to try to unify the different facets currently used to aggregate visualisations. Historically, geometric aggregation is very different from data aggregation and geographic aggregation. Unifying them would facilitate the software integration of components from the different domains.
- Allow deeper integration of all the visualisation domains. Most existing systems use side-by-side views, barely coordinated. Adapting existing coordination mechanisms to work with all the visualisation domains would facilitate linked and coordinated views.
- Improve research on software architectures for collaborative visualisations to allow the software infrastructures to be usable in single-user and multi-user settings.

6.3.2 Data Management

Data management model should provide distribution, in-memory caching, notification management and expressive typing

Since all the components of visual analytics require data to be stored and distributed to other software components, the data management component seems to be a good candidate to be the central mechanism for data and, to some extent, for distribution.

Information visualisation systems rely on an in-memory database to maintain their data. Relying on a solid database system would allow the domain of visual analytics to grow to larger sizes and lead to more robust applications.

Looking at the services described in Section 6.2.2, we can list the most important features that a successful data management architecture should provide:

Data Typing

The standard typing provided by SQL is not sufficient; higher-level types should be supported, in particular those listed by Card and Mackinlay[24]. At the infrastructure level, these types can be seen as metadata: there is a need to support rich metadata to adapt to rich information associated with the data. More sophisticated types should also be supported at the storage level. For example, there are several ways to aggregate numerical values – currently, most databases support single-valued summarisation, such as average or median, but more sophisticated summarisation include min-max or distribution histograms. Supporting these types, among others, is essential for analysis and visualisation. Special types have already been specified for geographical databases, it is important to allow these extensions at the database infrastructure level.

Managing and indexing dynamic, streamed data requires new mechanisms

Managing dynamic data, including streamed data, is also very important and not standard in databases. Time-stamped and volatile data is becoming increasingly important to manage. One of the difficult issues associated with this kind of data is in indexing and summarisation. Depending on the application domain, streaming data can be summarised with simple statistical values or more complex types such as wavelet transforms. Current databases do not support these types of analysis on the fly.

Distribution

Distribution is needed at several points of visual analytics systems; unify it when possible

Most databases are distributed using simple network connections. However, the performance of streamed-network links is low compared to the processing power of existing hardware architectures. Newer database systems offer datagram distribution for fast replication. Allowing more flexible and faster distribution protocols will allow the overall visual analytics infrastructure to grow to larger sizes and higher processing power. A fast distributed database can become the central point to manage distributed processing using newer parallel architectures, such as computer grids and multi-core GPUs, and distributed rendering systems, such as wall-sized displays, large tabletops or collaborative environments.

Distribution should also involve caching mechanisms so that the same software infrastructure can be used to manage massive databases and in-memory databases in a consistent way. Current visual analytics applications manage the transfer of relevant data in ad-hoc ways with little cooperation between the central database and the in-memory one, and no compatibility at the programming level.

Atomic Transactions

Visual analytics requires long transactions that are not supported by standard databases. Since analytical components may run for hours, days and weeks, the data manager needs to support very long commit phases, probably with some reconciliation strategy to deal with errors instead of promoting a complete fail. If analytical components can save partial results, they can finish transactions at a faster pace but it can take minutes or hours before a meaningful cycle of operation is ready to be committed. Traditional databases do not support these long transactions, although some drafts have been submitted for standardisation by major vendors using 'snapshot' isolation. More research work should be devoted to specifying a semantic of long transactions compatible with analysis, and to designing mechanisms for interactive refresh of visualised structures.

Notification

Notification in databases is currently implemented through the trigger mechanism, which executes some code when the data is modified. The support for triggers is very heterogeneous from one database to the other. While Oracle supports general triggers and extend them to notify on structural changes (schema modification), others such as MySQL lack much of this functionality.

These weaknesses hamper the use of standard databases for visual analytics and force practitioners to implement some kind of in-memory database that are certainly not as powerful and reliable as the mature database engines, but they fulfil the important requirements of visual analytics.

Revisiting database mechanisms such as notification will improve visual analytics

Newer database systems such as MonetDB offer a low-level layer of implementation where new kinds of notification mechanisms can be implemented and experimented with. The view of MonetDB as a 'memory shared across the network' instead of a facility to store and query data appears to be suited to visual analytics.

Interactive Performance

Visualisation systems and analytical systems need optimised in-memory data structures. They also implement the standard visualisation pipeline 'backwards', meaning that it is the view that triggers the computation of visible parts, pulling computations from the pipeline instead of just displaying the already computed contents of the previous stages. This is very different from what current analytical systems and databases provide.

Currently, database systems are not designed to allow fast in-memory computation or rendering. Visual analytics require high performance and so finding mechanisms to unify fast memory management with persistence, a fast query mechanism and distribution, would allow visual analytics to work

on a solid base. If this is not possible, then more work is required on a good separation of issues between database technologies and analysis and visualisation technologies, to avoid duplicating design and implementation efforts.

Computation

Current workflow systems connected to databases work by computing 'forward', starting from the beginning of the dependencies to the end. As mentioned above, visualisation systems usually work backward by pulling the required data from the pipeline, computing it on demand, steered by the analyst. Can workflow systems be improved to support this pull mechanism, allow some steering and to provide on-demand approximate solutions quickly to improve them later when possible?

Finding mechanisms and policies to allow large-scale asynchronous pull computation needs more research and experiments before it can be specified and standardised.

6.3.3 Analysis and Data Mining

The analysis and data mining domains use a simple, yet effective software architecture. Several implementations now work on local machines between different products, through the Web or on the Cloud. However, this architecture is not suited to visual analytics applications as it is. The main issues are:

- **Progressive analysis:** provide quick answers first, then make improvements incrementally or on-demand;
- **Management of dynamic data:** incremental analysis instead of restarting it from the beginning;
- **Steerable analysis:** allow long-computations to be steered by users when possible.

Currently, no data analysis tools provide these services. There are two paths that can be pursued to solve this gap: a) combine existing services to try to obtain the desired results or b) re-implement existing systems to provide the services.

As a first step, specifying a consensual software model for an analytical component will be required. All the analytical communities should be gathered to find agreement and social acceptance since this new software model will certainly require considerable work to be fully implemented and functional.

Conclusion

To better understand the interdisciplinary software architectural issues of visual analytics, all the specialists of toolkits and tools from the domains involved should meet and publish a white paper on recognised issues and ways to solve them. The VisMaster project has made a start by organising two workshops but the scope of the problem is so broad that it will need several more workshops, involving more focused domains, in order to move towards a good understanding of interdisciplinary issues and ways to implement them in a modular and extensible fashion.

The diversity of problems advocates for open standard rather than proprietary solutions

The diversity of problems addressed by the visual analytics community advocates for open standard rather than proprietary solutions. While some proprietary solutions are already available, most visual analytics applications will certainly need several analytical modules from several vendors and trying to monopolise the market with proprietary interfaces will instead slow-down the growth of the visual analytics field and delay the creation of a market for rich analytical components that can be integrated in interactive applications.

6.4 Opportunities

Solving the architecture issues will open a new market for components

Despite the large number of challenges facing the design and development of software infrastructures for visual analytics, the opportunities are considerable and viable. They are both scientific and commercial. Scientifically, the increased production of data should be harnessed but this requires new methods. However, even if the principles of exploring and analysing data become better understood every day, benefiting from visual analytics will require well designed software infrastructures.

Once these infrastructures are available, scientists and practitioners will devote fewer resources to specific software developments, instead they will rely on sound infrastructures to build their visual analytics applications.

Commercially, the market of visual analytics components does not exist because the requirements for these components are not well understood. However, the demand for such components is already high. Once the requirements and specifications of these components (or abstract components) are known, several companies, with varying sizes, will be able to provide their expertise and added value to practitioners in visual analytics and more generally to improve the usability of data-intensive applications.

When analytical components are usually small, there is also a need for larger systems such as databases and visualisation systems. New databases will be able to cope with massive data interactively. Addressing data management issues will certainly lead to new database engines, faster, more scalable, more distributed and providing analytical and interactive capabilities. New visualisation systems will be able to cope with very large datasets and to allow newer kinds of

visualisations, combining or merging scientific visualisation, geographic visualisation and information visualisation when needed.

6.5 Next Steps

The topic of software architecture for visual analytics is broad. To come to a common understanding of the problems and reach a consensus, we need contact points: workshops and conferences gathering all practitioners, research and industry. In the area of software infrastructures and standardisation, industry is often ahead of researchers.

Research agencies should create incentives for researchers in databases, analysis, visualisation and communication to work together to iterate on the design of a conceptual architecture for visual analytics. If this problem is not considered as a whole for all domains of visual analytics, as opposed to specific in each and every domain, we will see an explosion of partial solutions to the overall infrastructure problem with issues in interoperability. Therefore, there should be a coordinated action aimed at solving the problem overall.

In practical terms, this could be done by funding a few applied-research projects to design and experiment software architectures for visual analytics. The outcomes of these projects should be partly open since the goal is to promote interoperability and compatibility. Since software architecture is an interdisciplinary issue, it cannot be effectively addressed by several smaller funded projects, gathering some experts in software architecture of visual analytics. Initially, the problem should be addressed as widely as possible. Later on, more targeted groups will certainly address more focused issues but they need to be aware of the big picture first.

Once some level of agreement is reached, some organisation should be created to promote and manage the specifications. It can be done in the context of an existing organisation such as the Object Management Group (OMG) or independently as a visual analytics alliance. In any case, it should gather interested parties from industry and research to help specify 'standard' software architectures and APIs for a visual analytics module to interoperate with the right level of functionality and provide clear semantics.

Such a coordination will not only be beneficial to visual analytics but also to the related domains. Data management will provide the right services to scale, data analysis will be easier to integrate in more interactive environments and visualisation will be easier to deploy.