Obvious: a Meta-toolkit for information visualization toolkits used in Visual Analytics

Jean-Daniel Fekete[*] INRIA
Pierre-Luc Hemery[†] INRIA

ABSTRACT

Put an abstract here.

Index Terms: K.6.1 [Management of Computing and Information Systems]: Project and People Management—Life Cycle; K.7.m [The Computing Profession]: Miscellaneous—Ethics

1 INTRODUCTION

Over the past several years, we have seen the development of a variety of information visualization toolkits [4, 3, 5, 7]. The choice of one of this toolkit is an prominent step in a software development and can be confusing for visual analytics software developers. So, in this article, we propose and present a way to abstract and unify existing information visualization toolkits.

Historically, this proliferation of toolkits can be explained by the fact that each created toolkit addresses a specific problem and/or is designed for a specific domain. Thus, there is dispersion in terms of capabilities, since each toolkit has unique and interesting visualizations and techniques. For example, some toolkits offer several kinds of graph layouts [7] when others support very sophisticated coordinated views without specific graph capabilities [3].

That is why, visual analytics developers are almost immediately confronted with a crucial decision: the choice of the information visualization toolkit to use. Since, this choice imposes the data structure and then this data structure imposes capabilities and techniques designed for it. Thus, if a developer needs a technique available in other(s) toolkit(s), he has to implement it from scratch: it is a waste of time and a source of errors.

To address this problem, instead of creating another ultimate toolkit, after a worhshop gathering several major authors of toolkits [1], we specified and implemented a meta-toolkit named Obvious. By meta-toolkit, we mean a set of interfaces abstracting services provided by information visualization toolkits following the InfoVis reference model [2].

To provide evidence of the usefulness of Obvious, we have implemented, in Java, Obvious binding modules implementing interfaces and abstractions defined in the specification for several toolkits (Infovis toolkit [4], Prefuse [3] and Jung [5]). They have been used to build some proof-of-concepts examples combining different toolkits and also to create pieces of software used by a current research project [6]. During those developments, we have seen the information visualization community can also benefit from Obvious, since important design questions emerged, and their answers will improve Infovis reference model

In addition, obvious allows developers to eliminate the crucial choice of the toolkit and to avoid to rewrite existing functions. The following use case shows it is now possible to combine toolkits. For example, they can choose a data model from JUNG toolkit for a graph, then query it with prefuse predicates, use a layout introduced in Infovis toolkit to display it and still used network algorithms introduced in JUNG. With obvious, there are no more design restrictions imposed by an initial choice for developer.

[*]e-mail:Jean-Daniel.Fekete@inria.fr
[†]e-mail:Pierre-Luc.Hemery@inria.fr

2 RELATED WORK

The proliferation of information visualization toolkits have raised interest in creating abstractions to create them and share functionnalities. Thus, there have been several studies in software engineering to describe design patterns for toolkits and concrete implementations to propose generic uses and mechanics for toolkits.

When creating a software, design patterns are usefull for they are a good way to facilitate its construction and to make its architecture clearer, more abstract and extendable. That is why, the information visualization community has proposed several generic design patterns for toolkits and applications to facilitate developers work.

Design patterns have been introduced early in InfoVis with the InfoVis standard reference model. It derives from the standard Model-View-Controller model and separates an application in three parts : data model, visualization and view. This pattern has been widely used in almost every toolkits because it provides a simple efficient high level architecture.

Nevertheless, toolkits following this model have implemented the three parts in different way. For example, on one hand, the data model in InfoVis toolkit is column oriented, on the other, in Jung it is tuple oriented. That is why, Heer et Al have proposed twelve designs to describe common patterns and to facilitate software and toolkit creation. They describe several designs for the data model (column or tuple oriented), and solutions for visualization and view. However, it only regroups common and efficient design patterns for visual analytics but does not answer to the question of how combining existing techniques to simplify developer's choice during a software conception. Certainly, if the toolkit prefuse mostly follows all these patterns, others only partially apply them: those designs can not be directly used to extend and combine existing works.

Attemps have been made to address this problem. For example, the JUNG toolkit proposes a way to make a JUNG graph usable with prefuse. It is a one shot solution to the basic problem. Also, Borner has proposed an infrastructure to wrap, reuse and combine existing infovis components. In practice, there are a lot of components, but they have loose coupling and the infrastruture mainly translates data model into another to combine components. If it fits for static visualization with no interaction, it could difficulty work in visual analyctics since we want to avoid copying data models.

As a conclusion, there has been important work made to provide generic design patterns for information visualization. However, existing toolkits do not apply them in an uniform way creating useless complexity. Some initiatives to link toolkits have emerged but some of then are one shot attempts to bind two toolkits so are not generic enough or some are only dedicated to static visualizations and are not applicable in our field. Thus, there is a strong need to create a unified software infrastructure for visual analytics application. To take all those requirements into account, we propose a meta-toolkit introducing a unified software infrastructure for visual analytics.

3 DESIGN OF THE META-TOOLKIT

4 EXAMPLES

Put examples here.

5 CONCLUSION

Put conclusion here.

ACKNOWLEDGEMENTS

The authors wish to thank A, B, C. This work was supported in part by a grant from XYZ.

REFERENCES

[1] Vismaster workshop on visual analytics software architecture. http://code.google.com/p/obvious/wiki/Motivation, December 2008.

[2] J. Heer and M. Agrawala. Software design patterns for information visualization. *IEEE Trans. Vis. Comput. Graph.*, 12(5):853–860, 2006.

[3] J. Heer, S. K. Card, and J. A. Landay. prefuse: a toolkit for interactive information visualization. In *CHI*, pages 421–430, 2005.

[4] Jean-Daniel Fekete. The Infovis Toolkit. ACM Symposium on User Interface Software and Technology (UIST 2003) Conference Compendium, November 2003.

[5] J. O'Madadhain, D. Fisher, S. White, and Y.-B. Boey. The jung (java universal graph/network) framework. Technical report, UCI-ICS, 2003.

[6] Vronique Benzaken and Jean-Daniel Fekete and Pierre-Luc Hmery and Wael Khemiri and Ioana Manolescu. EdiFlow: data-intensive interactive workflows for visual analytics. In *International conference on Data Engineering (ICDE 2011)*, Hannover, Germany, 04 2011. IEEE. to appear.

[7] C. Weaver. Building highly-coordinated visualizations in improvise. In *INFOVIS*, pages 159–166, 2004.