End-User Development of Information Visualization

Kostas Pantazos[1], Soren Lauesen[2], and Ravi Vatrapu[1,3]

[1] Computational Social Science Laboratory (CSSL), Department of IT Management, Copenhagen Business School, Copenhagen, Denmark
[2] Software and Systems, IT University of Copenhagen, Copenhagen, Denmark
[3] Norwegian School of Information Technology, Oslo, Norway

Abstract. This paper investigates End-User Development of Information Visualization. More specifically, we investigated how existing visualization tools allow end-user developers to construct visualizations. End-user developers have some developing or scripting skills to perform relatively advanced tasks such as data manipulation, but no formal training in programming. 18 visualization tools were surveyed from an end-user developer perspective. The results of this survey study show that end-user developers need better tools to create and modify custom visualizations. A closer collaboration between End-User Development and Information Visualization researchers could contribute towards the development of better tools to support custom visualizations. In addition, as empirical evaluations of these tools are lacking both research communities should focus more on this aspect. The study serves as a starting point towards the engagement of end-user developers in visualization development.

Keywords: End-User Development, Information Visualization, Visualization Tools.

1 Introduction

Information Visualization attempts to reduce the time and the mental effort users need to analyze large datasets by visually presenting abstract data (e.g. medical information such as patient name, age, treatment, dose, intake, etc) that "has no inherent mapping to space" [1]. Unlike scientific visualization such as radiology, in information visualization there is no spatial correspondence between the physical information and the conceptual domain. Information Visualization is an important topic in many domains: clinicians want a complete picture of patient data; project managers need to obtain an overview and identify the bottlenecks in a project; database analysts look for visualizations that can locate trends in large datasets. Traditionally, visualization development is collaboration between domain experts and professional programmers. Both parties spend time and resources to design a good visualization. Usually, there are communication problems between users and programmers [2]; users have the domain knowledge

Y. Dittrich et al. (Eds.): IS-EUD 2013, LNCS 7897, pp. 104–119, 2013.

but no programming skills, while programmers do not have the domain expertise. Consequently, the process may require time and resources. From a management perspective, this collaboration can become very expensive. One solution to this problem would be to allow different domain users to construct visualizations. As a result, the cost would be significantly reduced, and better visualizations would be developed as users know their own domain-specific analytical needs and demands better.

In the last decade, a new research discipline has emerged, called End-User Development (EUD). EUD has its roots from the field of End-User Programming [3–5]. However, EUD is not limited to programming [6] and the main goal of EUD is to empower end-user developers – users who "may have little or no formal training or experience in programming" [7]– create, modify and extend *software artifacts*, and as a result gain more control over their applications by engaging them in the development process [8]. In 1995, Boehm et al. [9] estimated that by 2005, there would be 55 million end-user developers in the United States. In 2005, Scaffidi et al. [10] used and improved Boehm's method to estimate that in 2012 there will be 90 million end-user developers. They predicted that 55 million will be users of spreadsheets or databases. Some of end-user developers are: system administrators, interaction designers, teachers, accountants, health care workers, managers, etc.

This paper investigates End-User Development of Information Visualizations. More specifically, it investigates how end-user developers can create visualizations with existing visualization tools. We selected 18 Information Visualization tools form research and industry. The results of this study showed that end-user developers need more and better tools to create visualizations. Furthermore, the results of this study serve as a starting point in introducing End-User Development of Information Visualization. Also, the study aims at driving the attention of both communities towards research paths that may lead to the discovery of new development approach for end-user developers.

The remaining of this paper is structured as follows. Section 2 and 3 provide a summary of End-User Development and Information Visualization. Section 4 discusses the importance of users in visualization development. Section 5 presents a brief summary of 18 development tools from Information Visualization field focusing on how end-user developers can create visualizations. The paper concludes with a discussion of the limitations of the study and conclusions.

2 End-User Development

The End-User Development (EUD) field is a new research discipline, which has emerged from research in Human-Computer Interaction, Cognitive Science, Requirements Engineering, Software Engineering, CSCW, Artificial Intelligence, Information Systems, and the Psychology of Programming [11]. As a relatively young discipline, the field is not mature enough when it comes to definition, terminology, approaches and subject areas [11]. However, Lieberman et al. [8] defines EUD as "a set of methods, techniques, and tools that allow users of software systems, who are acting as non-professional software developers, at some

point create, modify and extend a software artifact". Consequently, end-user developers are not professional programmers, but users who "may have little or no formal training or experience in programming" [7]. EUD aims at better users efficiency and effectiveness as it allows users "to develop and evolve their computer based working tools to support their specific tasks in an efficient way" [12]. Therefore, the main goal of EUD is to empower these users create, modify and extend software artifacts, and as a result gain more control over their applications by engaging them in development.

EUD takes a broader perspective than End-User Programming because it is not limited to programming when it comes to adjust application to users' needs [6]. Lieberman et al. [8] defines two types of end-user activities: *parametrization or customization* (activities that allow end-users to parametrize or customize their applications using the available presentations or interactive mechanisms) and *program creation and modification* (activities that allow end-users to create or modify software artifacts). In order to support these types of end-user activities, the system should be flexible and expressive enough to changes (e.g. set parameters, compose objects, etc.) [8]. Simple changes are not difficult, but things become more complicated as the level of complexity for a change increases. MacLean et al. [13] suggested a "gentle slope" to reduce the level of complexity and support changes on different levels. However, in cases of extensive change actions a programming language should be used [13]. EUD does not focus only on how to support end-users create an application, but also focuses on the use and adaption of the application in existing environments [14]. The second means customizing, configuring and tailoring a application, but not direct changes in the source code [15]. Customizing, configuring and tailoring are performed beyond the stage of creating a new application, and take place after the application is implemented within its organizational infrastructure. Bolmsten and Dittrich [12] presented two case-studies and discussed the challenges that infrastructure context poses to EUD. In this study, we primarily focus on program creation and modification.

As there is not an EUD taxonomy which categorizes development techniques for end-user developers, several techniques developed from the psychology of programming are inherited, and some of the main techniques useful for end-user developers are [6]: Scripting Language, Visual Programming, Spreadsheet and Programming by Example.

Rode et al. [16] investigated EUD of web application. Their study showed that web development tools focus more at supporting developers with a wide range of functionalities, and less attention is paid to ease-of-use. Further, the authors say that ideally a web development tool "would provide ease-of-use with the appropriate abstractions, absence of jargon, a library of examples and templates, wizards for complicated tasks and take a holistic approach by integrating all aspects of web development" [16]. Investigating how end-users think may help in designing better tools. Similar to this study, we investigate visualization development tools.

3 Information Visualization

Information visualization (InfoVis) enhances human cognition by visually presenting abstract data and revealing patterns, trends and outliers [1]. The InfoVis field has enabled development of visualization systems that enhance human cognitive processes by visually presenting abstract data [1]. Although, the InfoVis field emerged during the 1980's with the availability of computers, InfoVis evidence can be tracked long before. Florence Nightingale's diagram designed in 1858 shows the death rates in the hospital of Scutari, and how the rates reduce by the changes introduced by nurse Florence Nightingale [17]. Thus visualizations (when designed by a domain expert) allows viewers to get a clear picture of the situation, and derive results without any detailed explanation.

The InfoVis community has done considerable work in order to develop the field into a mature discipline. Shneiderman [18] presented a task by data type taxonomy for InfoVis. This popular taxonomy classifies visualization data types (1D Linear, 2D Map, multidimensional, temporal, tree and network) and identifies the tasks (overview, zoom, filter, details-on-demand, relate, history and extract) that have to be supported. The reference InfoVis model described by Card et al. [1] highlights end-user interactions, and it consists of three steps: (1) *Data Transformations*: convert the raw data to data tables; (2) *Visual Mappings*: convert the data tables into visual objects; (3) *View Transformations*: transform visual objects into views by means of visual objects properties. The first step is mainly related with data, while the last two have a direct impact on the visual form.

Several visualizations have been developed to present data. LifeLines [19] is an interactive visualization that presents an overview of a patient's medical record. LifeLines 2 [20] and LifeFlow [21] are two other examples of temporal data. Aigner et al. [22] provide an overview of 101 visualizations techniques for temporal data. Among them are Arc Diagrams, Circel View, Circos, Flow Map, Prespective Wall, TimeTree, etc. Many of them have been developed in close collaboration with domain experts. This collaboration has contributed in producing useful visualizations.

4 Users in Visualization Development

Considering the variety of data and user tasks, it is obvious that new visualizations are needed. However, developing new visualizations is not an easy task. Several InfoVis toolkits and tools [23–30] have been developed to improve visualization development and provide better presentation of data. Providing good data visualization is challenging as visualization creators should have a good understanding of the data, and then properly design representations that allow users to accomplish tasks effectively and efficiently. This is usually a problem according to Thomas and Cook [31], who say that: "Most visualization software is developed with incomplete information about the data and tasks. New methods are needed for constructing visually based systems that simplify the development process and result in better targeted applications."

To facilitate the visualization development process and ensure that visualizations provide complete information about the data and tasks, several InfoVis applications (e.g. [19, 32–34]) have been developed applying the user-centered method, where users participated during the entire development process. Norman[35] and Nielsen[36] describe user-centered design as the early and continuous involvement of end-users in the design and development process. Considerable work has been conducted by Slocum et al. [32], Robinson et al. [33], Roth et al. [34], and Koh et al. [2] to define the activities applied in the user-centered model for the design and implementation of InfoVis tools. For example, Robinson et al. [33] describe a six-stage user-center design process (work domain analysis, conceptual development, prototyping, interaction and usability studies, implementation, and debugging) where users are involved and provide input in each stage. Using this model [33], Roth et al. [34] present a modified user-centered design approach, which starts with prototyping, followed by interaction and usability studies, work domain analysis, conceptual development, implementation and ends with debugging. Although, the user-centered model helps producing better visualizations, still it is challenging to bridge the gap of knowledge between end-users and programmers. This gap can influence communication and create challenges such as: programmers should understand end-user needs, end-users should gain some knowledge regarding InfoVis, end-users should be devoted and actively participate in the process, etc. In their study Koh et al. [2] experienced similar challenges where simple users where more interested in the tool than on questions about their tasks and data. Also, when they tried the tool they found it limited compared to the prototypes defined during the process. The authors [2] suggested that an iterative approach may address these issues.

Although a user-centered method is a successful approach, researchers envisage approaches to facilitate visualization development and assure that visualizations provide complete information about the data and tasks. Aigner et al. [37] discuss how to support user-centered visual analysis that consists of three factors: the visualization, the analysis, and the user. They suggest that future research should focus on these three factors and lead to the convergence of user-centered visual analysis. Their vision matches the universal usability challenge defined by Plaisant [38]. According to Plaisant [38] visualization tools should be accessible to diverse users that do not have the same background, technical knowledge, or personal abilities. Other InfoVis researchers seek ways of introducing new audiences in InfoVis. Heer et al. [39] say that designing visualizations is not an easy task for users, but "we have to provide them tools that make it easy to create and deploy visualizations of their datasets" [39].

5 InfoVis Development Tools - A Survey

The purpose of this survey is to investigate how end-user developers are supported by InfoVis tools in visualization development. To the best of our knowledge, no prior study has looked at EUD of InfoVis. The results of this survey

may serve as a starting point towards the engagement of end-user developers in visualization development. Before we present the tools, we describe the tool selection process and how the tools were assessed. The purpose of this study is not to analyze and compare implementation details, but to investigate the way end-user developers construct visualizations. For a deeper understanding of implementation details we encourage readers to refer to the references.

5.1 Analysis Approach

We used two professional and popular sources to find InfoVis tools and toolkits: the ACM Portal and the IEEE website. We searched for related work by combining these keywords: "information visualization", "tool", "toolkits", "graphical user interface". Initially, we ranked the results based on the total number of citations, and then we selected only the most relevant ones by reading the abstracts. Next, we read all the papers and selected the most appropriate tools and toolkits. They are: APT [40], SAGE & SageBrush [41–43], DEVise [44, 45], The InfoVis Toolkit [23], GeoVISTA Studio [46], Piccolo [47], Improvise [48], Prefuse [24], Protovis and ProtoViewer [49, 50], and Data-Driven Documents (D3) [25]. During the process of reviewing the existing literature, we identified two more tools from research that were relevant to the investigation and decided to include them in the analysis, because of their popularity and approach. They are: Processing [51] and Flare [29]. In total, we selected 12 tools from the research area. As we were reviewing the existing literature, we also found several industry tools that we decided to use. At the end we selected six popular tools: Spotfire [26], Tableau [27], Omniscope [28], MS Excel [30], Google Chart Tools [52] and Many Eyes [53]. In total, we chose only 18 tools and toolkits and we believe that the selected ones are a good sample that represents the wide-range of InfoVis tools from research and industry.

In this study, we investigated how end-user developers can construct visualizations with existing development tools. We conducted our tool analysis focusing on three main questions:

1. Can end-user developers create and modify a visualization?
2. How do end-user developers create and modify visualizations with a tool; Do they specify language specifications (e.g. Java, JavaScript, etc.), use wizards or drag-and-drop actions?
3. Can tools support development of predefined and custom (not-predefined) visualizations? A predefined visualization, for instance a bar chart in MS Excel, uses a chart where only a few visual attributes can be assigned to data. While LifeLines [19], a custom visualization, combines bars, triangles, labels, etc., into a complex visualization.

Investigating these questions will provide an overview of the current status of InfoVis development tools and reveal their accessibility to end-user developers. The assessment of the tools from academia is based on the published papers. The commercial tools were assessed using the trial or the full versions, and information from their websites. A full-fledged usability study is currently scheduled for Fall 2013 and will be reported in subsequent publications.

5.2 Tools and Toolkits

In this section, we briefly describe the selected tools. First, we present InfoVis tools and toolkits from research, and then the ones from industry.

APT (A Presentation Tool) [40] is one of the earliest tools that automatically creates effective graphical presentation of relational data. Presentations are generated in a linear model where data are extracted, synthesized and then the tool handles the rendering process to create the final output. Users of APT use predefined visual objects (e.g. bar charts, scatter plots or connected graphs) and write their graphical specifications (sentences of a graphical language that has exact syntax and semantics), and the tool creates the graphical presentation. The visual mapping is defined through APT specifications and automatically handled by the tool. Probably, end-user developers, would be able to specify graphical designs, but still they cannot create visualizations other than the supported ones.

SAGE & SageBrush: Early 1990's, Roth and Mattis [41] presented SAGE, "an intelligent system which assumes presentation responsibilities for other systems by automatically creating graphical displays which presents the results they generate" [41]. This tool uses graphical techniques to express the application data characteristics and fulfill the presentation needs. Users of SAGE query the database, and the result is used by SAGE. Based on the data, SAGE automatically defines the visual mappings and generates the visualization. After a presentation is generated, users can adjust the visual mappings of the auto-generated visualization by setting layout constraints for the data. SAGE can probably be used by end-user developers.

SAGE was extended with an interactive design tool called SageBrush [42, 43]. SageBrush aims at removing the complexity introduced by SAGE representations and operations [43]. It allows users to sketch by dragging and dropping primitives or partial controls from a palette. The sketches are used by SAGE to create a visualization. SageBrush facilitates visualization development and can be used by end-user developers. They can create predefined and custom visualizations with drag-and-drop actions.

DEVise [44, 45] allows users to create visualizations by creating, modifying or connecting visual objects. DEVise maps the data to visual objects and displays them in a view. At the end, the view uses the data and visual filters to draw the result in a window. DEVise users use a step-by-step approach to create visualizations: select an input, choose a file type for the input file, select an existing mapping or define a new mapping using *tcl* language expressions [54], select a view to display the data, select initial values for the visual filter, and finally select a window to display the view. In DEVise, end-user developers can create custom visualizations by combining and linking visual objects using the predefined visual mappings. In order to create new visual mappings, they have to use the *tlc* language. The authors says that DEVise is a powerful exploration

framework, "but to appreciate this power fully, one must work with the system or at least look at several applications in some details" [45].

Processing was developed initially "to teach fundamentals of computer programming within a visual context" to newcomers, but it has grown into a more complete tool for constructing images, animations and interactions [51]. Processing has a development environment similar to a regular one. Programmers specify visual mappings by writing code in the code editor. They view the visualization in a new window after having executed the code. To create predefined and custom visualization, users have to know a programming language called Processing. This tool cannot be used by end-user developers via direct manipulation in the WYSIWYG (What You See Is What You Get) paradigm.

GeoVISTA Studio is a development environment designed to support geoscientific data analysis and visualizations [46]. It is built in Java and uses JavaBeans technology. A visualization in GeoVISTA Studio is composed by connecting visual objects (implemented as Java beans components). GeoVISTA Studio consists of three windows: the *Main* window shows the menus and JavaBeans visual object palette; the *Design* window where visual objects are placed and connected; the *Graphical User Interface (GUI)* window shows "live" the output of the used beans. Programmers can use the *Property Editor* to customize the appearance and behavior of a visual object. The application programmers (probably end-user developers) are the main users of the Studio, and they follow the following steps to construct an application: list the requirements, select the appropriate visual objects from the palette menu (new visual objects can be developed outside of the Studio and imported), place visual objects in the *Design*, link the visual objects to meet the requirements, customize a visual object using the *Property Editor*, and test the design in the *GUI*.

The InfoVis Toolkit [23] is a Java based visualization toolkit that uses several interactive controls to construct visualizations. This toolkit allows programmers to program visualizations. It allows programmers to extend the toolkit with new controls and to integrate visualization techniques into interactive applications. However, creating visualizations requires experienced programmers. Consequently, this toolkit is not appropriate for end-user developers.

Piccolo [47] is mainly used for developing graphical applications with rich user interfaces. It is developed in Java and C# and supports the development of visualizations indirectly, as it does not support visualization techniques [24]. Nevertheless, novel visualizations are based on this toolkit. Programmers can create visualizations in Java or C# and use visualization functionality and controls, such as zooming, animation and range slider. This toolkit can be used only by programmers, and the fact that it does not support visualization techniques directly, makes it challenging even for them. End-user developers cannot use this tool.

Improvise [48] is a visualization toolkit for creating multi-view coordination visualizations for relational data. It is written in Java. Visualizations are created by specifying expressions for simple shared-object coordination mechanism. Shared-objects in Improvise, which are responsible for visual mappings, are graphical attributes such as color, font, etc. Improvise has a specialized development environment where users apply a step-by-step approach interacting with four editors and creating views by adding frames, controls, defining variables and attaching data using the lexicon work area (a central repository where information related to the data and database are saved). Users of Improvise can construct visualizations based on the predefined controls. Programmers create visualizations by specifying expressions for simple shared-object coordination mechanism. Although we believe that Improvise can be used by end-user developers, this has not been empirically evaluated.

Prefuse [24] is another toolkit developed in Java. Visualizations in Prefuse are programmed in Java. Programmers construct them using a set of fine-grained building blocks and specifying operators that define the layout and behavior of these blocks. The purpose of this tool is to facilitate programmers' work, but end-user developers cannot use this toolkit.

Flare [29] is a successor of Prefuse [24], but is written in ActionScript. Flare supports programmers develop visualizations. To construct visualizations, programmers specify in ActionScript the properties of the visual objects and sequential commands. Programmers can also define new operators and visual objects, but advanced programming knowledge is required. Flare cannot be used by end-user developers.

Protovis & ProtoViewer: Protovis [49] is implemented in JavaScript and helps programmers construct visualizations using a domain specific language. They can combine primitive visual objects, called marks, bind them to data, and specify visual properties. Programmers can create visualizations by specifying Protovis specifications. The authors of Protovis have compared the specifications for a simple pie chart in Protovis, Processing and Flare, showing that the visualization in Protovis is specified in fewer lines of code [49]. This shows the simplicity of Protovis language, which has a high potential of engaging end-user developers in visualization development. Although we believe that Protovis can be used by end-user developers, there is no empirical evidence that proves it.

ProtoViewer [50] extends Protovis with a development environment. The screen is divided in three parts: *Data*, *Design* and *Code*. Programmers choose a dataset, select a visualization template and automatically the code is shown in the *Code* editor. They execute the code to view the results in the *Design*. Programmers can either use predefined visualization templates, and the code is automatically shown in the *Code* editor, or start from scratch and write Protovis specifications to specify controls. Constructing custom visualizations by end-user developers in Protovis becomes even more realistic by means of its

development environment – ProtoViewer. However, neither Protovis nor ProtoViewer has been evaluated with end-user developers.

Data-Driven Documents (D3) [25] is a successor of Protovis [49]. Visualizations are constructed using SVG, HTML 5 and CSS. In D3 the data transformation, the immediate evaluation and the browser's native representation are handled in more effective and transparent way than Protovis, which uses more succinct specification for static presentations [25]. However, these improvements introduce an overhead for users: the knowledge of SVG, HTML 5 and CSS. This toolkit is not suitable for end-user developers as it requires advanced programming skills.

MS Excel [30] is a spreadsheet program that allows end-user developers to analyze and visualize data. With simple steps, end-user developers can construct visualizations based on predefined visualization templates (e.g. bar chart, pie chart, etc.) They select a visualization template (e.g. bar-chart) and specify spreadsheet formulas or use standard wizards to map the data to the visual object in the worksheet area. In MS Excel, visual mappings are limited and end-user developers can set only predefined visual properties.

Tableau [27] is a commercial visualization tool, a successor of Polaris [55] developed at Stanford University. Tableau allows end-user developers to construct visualizations by dragging and dropping fields onto axis shelves (vertical and horizontal areas) and using visual specifications. This tool provides drag-and-drop features and several wizards to facilitate development. Further, it has a powerful interactive development environment where end-user developers can interact, filter, sort data and create interactive dashboards. Tableau is a "black box" system and constructing visualizations other than the predefined ones is not possible.

Spotfire [26] is another commercial tool for data visualizations. It supports end-user developers with a number of visualization techniques. End-user developers interact with the development environment and construct visualizations based on predefined ones. Once they select the data and choose a visualization template, the tool automatically generates the visualization. Users can sort, filter and rearrange data by simply dragging and dropping fields in the design area. Users can also create dashboards, by combining different predefined visualizations (e.g. bar chart, scatter plot, etc.) in a single screen. As in Tableau, end-user developers can only create predefined visualizations.

Omniscope [28] is in the same category as Tableau and Spotfire, and shares similar features such as interactive dashboard, drag and drop features, etc. It supports end-user developers in constructing predefined visualizations, as Spotfire and Tableau do. Custom visualizations cannot be constructed with this tool.

Google Chart Tools [52] is a library written in JavaScript that provides several predefined simple (line chart, scatter chart, etc.) and advanced chart types (Image multi-color bar chart, Motion Chart Time Formats, etc.) Visualizations can be constructed by end-user developers in the web-based development environment named Code Playground. In addition, Google Chart Tools has another environment named Live Chart Playground, to test charts already created in the Code Playground. In Live Chart Playground, end-user developers can change some parameters and see how the visualization changes. End-user developers are limited to predefined visualizations.

Many Eyes [53], developed at IBM Research Center, is a web-based visualization platform that can be used by end-user developers. In Many Eyes, visualizations are implemented in Java Applets. End-user developers construct visualizations in three steps: upload a dataset; choose a visualization template; customize and publish the visualization. Many Eyes automatically generates and shows the visualization on the screen. Custom visualizations are not supported.

Results

In this study, we surveyed 18 visualization tools from an end-user developer perspective. Based on published papers and subjective evaluation of the selected the tools, we found 12 InfoVis tools that have the potential to be used by end-user developers. 11 tools allow end-user developers to construct visualizations with a programming language (e.g. Java, ActionScript, JavaScript, etc.), and six with a wizard or drag-and-drop actions. Furthermore, 11 tools support development of predefined and custom (not predefined) visualizations, but only five of them can be used by end-user developers. Figure 1 provides an overview of the results.

There is a tendency that researchers mainly focus on developing visualization tools that allows users to construct predefined and custom visualizations, but users need advanced programming skills. End-user developers would not be able to benefit from these tools. Some examples are: Prefuse, Flare, D3, etc. On the other hand, industry produce visualization tools for large audiences without advanced programming skills, but at the same time limit them with predefined visualization templates. Although both communities can benefit from the engagement of end-user developers in constructing custom visualizations, they are overlooked. Only five visualization tools (SAGE/SageBrush, DEVise, GeoVISTA Studio, Improvise and Protovis/ Protoviewer) may support them in constructing visualizations other than predefined. To the best of our knowledge, none of the selected tools were empirically evaluated with potential users. As a result, it remains debatable if the five tools can support end-user developers in visualization development. This indicates that InfoVis community has to focus more on evaluation of development tools with users. In addition, a future collaboration between EUD and InfoVis researcher may address this issue and lead to better tools for the advancement of both communities.

The results also show that commercial tool provide interactive development environments where users can use wizard and/or drag-and-drop actions. These

Tools	End-User Developer	Visualization Development	Predefined Visualizations	Custom Visualizations
Processing		Programming Language	x	x
InfoVis Toolkit		Programming Language	x	x
Piccolo		Programming Language	x	x
Improvise	x	Programming Language	x	x
Prefuse		Programming Language	x	x
Flare		Programming Language	x	x
D3		Programming Language	x	x
SAGE / SageBrush	x	Programming Language & Drag-and-Drop Approach	x	x
Protovis / Protoviewer	x	Programming Language & Wizard Approach	x	x
GeoVISTA Studio	x	Wizard & Drag-and-Drop Approach	x	x
DEVise	x	Wizard Approach	x	x
ATP	x	Programming Language	x	
Google Chart Tools	x	Programming Language	x	
Tableau	x	Wizard & Drag-and-Drop Approach	x	
Spotfire	x	Wizard & Drag-and-Drop Approach	x	
Omniscope	x	Wizard & Drag-and-Drop Approach	x	
MS Excel	x	Wizard Approach	x	
Many Eyes	x	Wizard Approach	x	

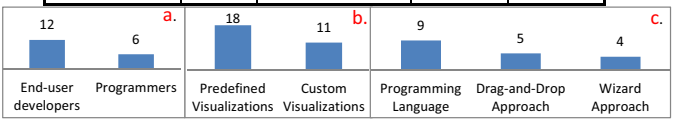

Fig. 1. 18 InfoVis surveyed from an end-user developer perspective. Classification by: a. end-user developer and programmers, b. predefined and custom visualizations, and c. visualization development approach.

environments aim at handling the gulf of execution (*How do I do something?*) and evaluation (*What happened?*) identified by Norman [35] by allowing users to easily map data to visual objects and obtain immediate feedback.

Custom visualizations in Prefuse, InfoVis Toolkit, D3, Flare, Processing and Picolo are created and modified through code. This makes them less accessible to end-user developers. Improvise and DEVise use a step-by-step approach to lower the barriers to development introduced by code and become accessible by end-user developers. While, SageBrush and GeoVISTA Studio take a different approach. Similar to commercial tools, in these two tools, end-user developers interact with visual components using drag-and-drop actions.

6 Limitations

This study investigates 18 visualization development tools. Instead of all existing InfoVis tools, we decided to include only 18 tools as that are representative of the InfoVis field and that have contributed significantly to it.

To identify InfoVis tools we searched two popular and comprehensive professional sources IEEE and ACM. InfoVis tools published in other sources such

as Springer, Elsevier, Sage, etc., were not included. As a result, the findings of this study may be debatable as there might be other tools published in these sources for end-user developers. Another limitation of the selection is that we investigated tools published before 2012.

Our investigation was based on the published papers and subjective evaluation of the selected the tools. However, we do not have the knowledge that authors of these tools have. This should have facilitated the analysis process. Furthermore, a task-based evaluation with end-user developers would have enriched the results of this study, which, however, provides a first orientation towards what tools might be suitable for visualization development. Also, this study does not investigate visualization and interaction techniques a tool may support.

7 Conclusion

This paper presents a study that investigates EUD of InfoVis. We investigated how existing InfoVis tools can support end-user developers create and modify visualizations. The results of this study indicate that EUD and InfoVis community has to focus more on developing new approaches and tools to allow end-user developers create visualizations other than the predefined ones. Supporting them with more tools that provide direct manipulation, immediate feedback may be a potential research path. Furthermore, the study provides a high-level overview of the available visualization tools, which may facilitate the tool selection process for new audiences.

To the best of our knowledge, no tool has been empirically evaluated with users. Therefore, both research communities should collaborate more on this aspect in order to better address the ease-of-use and understand what makes visualization tool popular for end-user developers. In this respect, we are planning to conduct task-based usability studies and evaluate InfoVis tools with end-user developers.

References

1. Card, S.K., Mackinlay, J.D., Shneiderman, B. (eds.): Readings in information visualization: using vision to think. Morgan Kaufmann Publishers Inc., San Francisco (1999)
2. Koh, L.C., Slingsby, A., Dykes, J., Kam, T.S.: Developing and applying a user-centered model for the design and implementation of information visualization tools. In: 2011 15th International Conference on Information Visualisation, pp. 90–95 (2011)
3. Nardi, B.A.: A small matter of programming: perspectives on end user computing. MIT Press, Cambridge (1993)
4. Cypher, A., Halbert, D.C., Kurlander, D., Lieberman, H., Maulsby, D., Myers, B.A., Turransky, A. (eds.): Watch what I do: programming by demonstration. MIT Press, Cambridge (1993)
5. Lieberman, H.: Your wish is my command: programming by example. Morgan Kaufmann Publishers Inc., San Francisco (2001)

6. Lieberman, H., Paternò, F., Wulf, V.: End User Development (Human-Computer Interaction Series). Springer-Verlag New York, Inc., Secaucus (2006)
7. Pane, J., Myers, B.: More natural programming languages and environments. In: Lieberman, H., Paternò, F., Wulf, V. (eds.) End User Development. Human-Computer Interaction Series, vol. 9, pp. 31–50. Springer, Netherlands (2006)
8. Lieberman, H., Paternò, F., Klann, M., Wulf, V.: End-user development: An emerging paradigm. In: Lieberman, H., Paternò, F., Wulf, V. (eds.) End User Development. Human-Computer Interaction Series, vol. 9, pp. 1–8. Springer, Netherlands (2006)
9. Boehm, B., Clark, B., Horowitz, E., Westland, C., Madachy, R., Selby, R.: Cost models for future software life cycle processes: Cocomo 2.0. Annals of Software Engineering, 57–94 (1995)
10. Scaffidi, C., Shaw, M., Myers, B.: Estimating the numbers of end users and end user programmers. In: Proceedings of the 2005 IEEE Symposium on Visual Languages and Human-Centric Computing, VLHCC 2005, pp. 207–214. IEEE Computer Society, Washington, DC (2005)
11. Klann, M., Paternò, F., Wulf, V.: Future perspectives in end-user development. In: Lieberman, H., Paternò, F., Wulf, V. (eds.) End User Development. Human-Computer Interaction Series, vol. 9, pp. 475–486. Springer, Netherlands (2006)
12. Bolmsten, J., Dittrich, Y.: Infrastructuring when you don't – end-user development and organizational infrastructure. In: Costabile, M.F., Dittrich, Y., Fischer, G., Piccinno, A. (eds.) IS-EUD 2011. LNCS, vol. 6654, pp. 139–154. Springer, Heidelberg (2011)
13. MacLean, A., Carter, K., Lövstrand, L., Moran, T.: User-tailorable systems: pressing the issues with buttons. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI 1990, pp. 175–182. ACM, New York (1990)
14. Dittrich, Y., Lindeberg, O., Lundberg, L.: End-user development as adaptive maintenance. In: Lieberman, H., Paternò, F., Wulf, V. (eds.) End User Development. Human-Computer Interaction Series, vol. 9, pp. 295–313. Springer, Netherlands (2006)
15. Ko, A.J., Abraham, R., Beckwith, L., Blackwell, A., Burnett, M., Erwig, M., Scaffidi, C., Lawrance, J., Lieberman, H., Myers, B., Rosson, M.B., Rothermel, G., Shaw, M., Wiedenbeck, S.: The state of the art in end-user software engineering. ACM Comput. Surv. 43(3), 21:1–21:44 (2011)
16. Rode, J., Rosson, M.B., Quinones, M.A.P.: End user development of web applications. In: Lieberman, H., Paternò, F., Wulf, V. (eds.) End User Development. Human-Computer Interaction Series, vol. 9. Springer, Netherlands (2006)
17. Spence, R.: Information Visualization: Design for Interaction, 2nd edn. Prentice-Hall, Inc., Upper Saddle River (2007)
18. Shneiderman, B.: The eyes have it: A task by data type taxonomy for information visualizations. In: Proceedings of the 1996 IEEE Symposium on Visual Languages, VL 1996, pp. 336–343. IEEE Computer Society, Washington, DC (1996)
19. Plaisant, C., Mushlin, R., Snyder, A., Li, J., Heller, D., Shneiderman, B., Colorado, K.P.: Lifelines: Using visualization to enhance navigation and analysis of patient records. In: Proceedings of the 1998 American Medical Informatic Association Annual Fall Symposium, pp. 76–80 (1998)
20. Wang, T.D., Plaisant, C., Quinn, A.J., Stanchak, R., Murphy, S., Shneiderman, B.: Aligning temporal data by sentinel events: discovering patterns in electronic health records. In: Proceedings of the Twenty-Sixth Annual SIGCHI Conference on Human Factors in Computing Systems, CHI 2008, pp. 457–466. ACM (2008)

21. Wongsuphasawat, K., Guerra Gómez, J.A., Plaisant, C., Wang, T.D., Taieb-Maimon, M., Shneiderman, B.: Lifeflow: visualizing an overview of event sequences. In: Proceedings of the 2011 Annual Conference on Human Factors in Computing Systems, CHI 2011, pp. 1747–1756. ACM (2011)
22. Aigner, W., Miksch, S., Schumann, H., Tominski, C.: Visualization of Time-Oriented Data, 1st edn. Springer Publishing Company, Incorporated (2011)
23. Fekete, J.D.: The infovis toolkit. In: Proceedings of the IEEE Symposium on Information Vizualization 2004, pp. 167–174 (2004)
24. Heer, J., Card, S.K., Landay, J.A.: prefuse: a toolkit for interactive information visualization. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI 2005, pp. 421–430. ACM (2005)
25. Bostock, M., Ogievetsky, V., Heer, J.: D3 data-driven documents. IEEE Transactions on Visualization and Computer Graphics 17(12), 2301–2309 (2011)
26. Spotfire, `http://spotfire.tibco.com/` (accessed August 2011)
27. Tableau, `http://www.tableausoftware.com/` (accessed August 2011)
28. Omniscope, `http://www.visokio.com/` (accessed August 2011)
29. Flare, `http://flare.prefuse.org/` (accessed August 2011)
30. Microsoft Excel, `http://office.microsoft.com/en-us/excel/` (accessed August 2011)
31. Thomas, J.J., Cook, K.A.: A visual analytics agenda. IEEE Comput. Graph. Appl. 26(1), 10–13 (2006)
32. Slocum, T.A., Cliburn, D.C., Feddema, J.J., Miller, J.R.: Evaluating the Usability of a Tool for Visualizing the Uncertainty of the Future Global Water Balance. Cartography and Geographic Information Science, 299–317 (October 2003)
33. Robinson, A.C., Chen, J., Lengerich, E.J., Meyer, H.G., MacEachren, A.M.: Combining usability techniques to design geovisualization tools for epidemiology. Cartography and Geographic Information Science 32(4), 243–255 (2005)
34. Roth, R., Ross, K., Finch, B., Luo, W., MacEachren, A.: A user-centered approach for designing and developing spatiotemporal crime analysis tools. In: GIScience 2010 (2010)
35. Norman, D.A.: The Design of Everyday Things. Doubleday Business (1990)
36. Nielsen, J.: Usability Engineering. Morgan Kaufmann Publishers Inc., San Francisco (1993)
37. Aigner, W., Miksch, S., Müller, W., Schumann, H., Tominski, C.: Visual methods for analyzing time-oriented data (January 2008)
38. Plaisant, C.: The challenge of information visualization evaluation. In: Proceedings of the Working Conference on Advanced Visual Interfaces, AVI 2004, pp. 109–116. ACM (2004)
39. Heer, J., van Ham, F., Carpendale, S., Weaver, C., Isenberg, P.: Creation and collaboration: Engaging new audiences for information visualization. In: Kerren, A., Stasko, J.T., Fekete, J.-D., North, C. (eds.) Information Visualization. LNCS, vol. 4950, pp. 92–133. Springer, Heidelberg (2008)
40. Mackinlay, J.: Automating the design of graphical presentations of relational information. ACM Trans. Graph. 5(2), 110–141 (1986)
41. Roth, S.F., Mattis, J.: Automating the presentation of information (1991)
42. Roth, S.F., Kolojejchick, J., Mattis, J., Chuah, M.C.: Sagetools: an intelligent environment for sketching, browsing, and customizing data-graphics. In: Conference Companion on Human Factors in Computing Systems, CHI 1995, pp. 409–410. ACM (1995)
43. Chuah, M.C., Roth, S.F., Kerpedjiev, S.: Intelligent multimedia information retrieval, pp. 83–111. MIT Press (1997)

44. Cheng, M., Livny, M., Ramakrishnan, R.: Visual analysis of stream data. In: Proceedings of SPIE/The International Society for Optical Engineering, vol. 2410, pp. 108–119 (1995)
45. Livny, M., Ramakrishnan, R., Beyer, K., Chen, G., Donjerkovic, D., Lawande, S., Myllymaki, J., Wenger, K.: Devise: integrated querying and visual exploration of large datasets. In: Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data, SIGMOD 1997, pp. 301–312. ACM (1997)
46. Takatsuka, M., Gahegan, M.: Geovista studio: a codeless visual programming environment for geoscientific data analysis and visualization. Comput. Geosci. 28(10), 1131–1144 (2002)
47. Bederson, B.B., Grosjean, J., Meyer, J.: Toolkit design for interactive structured graphics. IEEE Trans. Softw. Eng. 30, 535–546 (2004)
48. Weaver, C.: Building highly-coordinated visualizations in improvise. In: Proceedings of the IEEE Symposium on Information Visualization, pp. 159–166. IEEE Computer Society (2004)
49. Bostock, M., Heer, J.: Protovis: A graphical toolkit for visualization. IEEE Transactions on Visualization and Computer Graphics 15(6), 1121–1128 (2009)
50. Akasaka, R.: Protoviewer: a web-based visual design environment for protovis. In: ACM SIGGRAPH 2011 Posters, SIGGRAPH 2011, p. 85:1. ACM (2011)
51. Processing, `http://www.processing.com/` (accessed August 2011)
52. GOOGLE CHART TOOLS, `http://code.google.com/apis/chart/` (accessed October 2011)
53. Viegas, F.B., Wattenberg, M., van Ham, F., Kriss, J., McKeon, M.: Manyeyes: a site for visualization at internet scale. IEEE Transactions on Visualization and Computer Graphics 13, 1121–1128 (2007)
54. Welch, B.B.: Practical programming in Tcl and Tk, 2nd edn. Prentice-Hall, Inc., Upper Saddle River (1997)
55. Stolte, C., Hanrahan, P.: Polaris: a system for query, analysis and visualization of multi-dimensional relational databases. In: IEEE Symposium on Information Visualization, InfoVis 2000, pp. 5–14 (2000)