PAVIS – Pervasive Adaptive Visualization and Interaction Service

Javid Alimohideen, Luc Renambot, Jason Leigh, Andrew Johnson
Electronic Visualization Laboratory
Robert L. Grossman, Michal Sabala
Laboratory for Advanced Computing
University of Illinois at Chicago
Chicago, IL 60607

Abstract
Prior research efforts have focused on the development of techniques for rule-based automated generation of either visualizations or user-interfaces. There is, however, little focus on adaptive visualization and interaction to provide a system that is capable of dealing with complex visualization and automated user-interface generation simultaneously. In this paper, we present our objectives in the development of new system so called PAVIS – Pervasive Adaptive Visualization and Interaction Service, an adaptive system that intends to automatically generate visualization and context-sensitive user-interfaces that are best suited for the display device. Based on the user's environment, display devices may vary widely from cell-phones, desktop computers to high-resolution tiled displays. A context-sensitive environment can respond to changes in visualization or data context by accommodating techniques to change the interface at run-time. Here, we describe our design goals and an initial case study.

Keywords: Adaptive Visualization, Dynamic User-Interfaces, Display Resolution, Scalable Rendering.

1. Introduction

Visualization provides a powerful mechanism for assisting in the interpretation of complex data. The form the visualization takes can therefore have dramatic impact on how effectively the data is interpreted [4]. Substantial advances in computing, networking and displays have enabled significant advances in our ability to process and visualize large volumes of data. However, our desire to access these visualizations have also increased. Users are no longer satisfied uniquely with desktop displays. They would like to be able to see and interact with the visualizations on portable devices such as cell-phones, PDAs, laptops, as well as large ultra-high-resolution tiled displays driven by clusters of computers. This is an extremely challenging problem for the application developer because expertise in program development, system architecture, visualization and user-interface design is needed to enable a successful "port" of the application to these various display systems. A trivial approach would be to constrain the solution to one that meets the lowest common capabilities of all the systems (e.g. an image on a web-browser.) A more intelligent approach that would bring about greater benefits would be to consider the limitations and capabilities of each class of visualization system and produce visualization and user-interface that takes special advantage of those capabilities. Recognition of these issues has given rise to work in automated generation of graphical representations and user-interfaces [4, 5].

In this paper, we present PAVIS – the Pervasive Adaptive Visualization and Interaction Service, a rule-based system that intends to automatically generate visualization and user-interfaces that are best suited for the display device. The scenario presented in this paper uses the Pantheon Gateway Testbed [10] as its main source of data. The US Army's Pantheon Project seeks to develop a system that processes real-time sensor data using data-mining algorithms to predict impending events and to distribute alerts to troops in the field as well as analysts and high-level decision makers. The Pantheon Gateway Testbed enables the testing of developed techniques by using non-classified data from the 830 highway traffic sensors in the city of Chicago, including weather data and text messages about events that might affect traffic [10]. While the developed techniques are fundamentally targeted for the Army, the testbed can provide immediate benefits for everyday citizens. For example, this real-time information could be useful to a person driving on an expressway who needs to obtain an estimate on his travel time to a destination; or it can predict when the city should mobilize snowplows in anticipation of an incoming storm.

2. Related Work

Rule-based techniques help modern visualization software systems support a wide variety of choices for mapping, manipulating and rendering data [1]. Rules define high-level policies and are used to automatically select and tune the visualization routines based on application requirements and available resources such as computing and networking [2]. A vast majority of research has been done on identifying several rule-based techniques for automated visualization and user-interface generation. In Rogowitz and Trenish [1], an architecture for

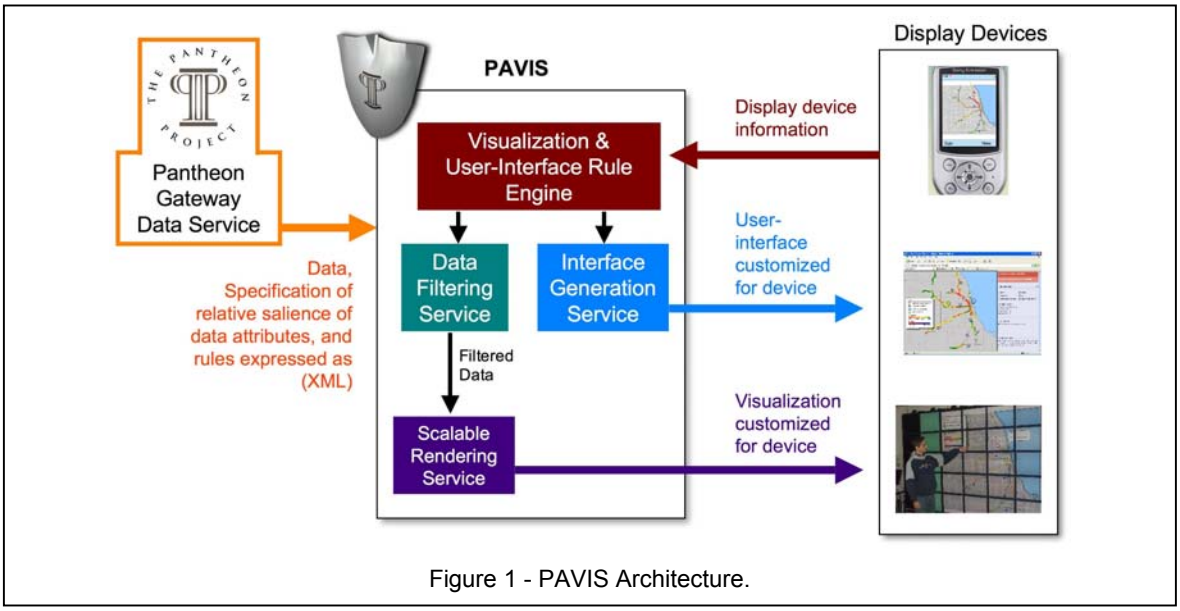

Figure 1 - PAVIS Architecture.

incorporating perceptual rules into the visualization process was introduced. A rule-based visualization sub-system to improve the flexibility of wide-area network-based computational steering collaboratory was developed in [2]. Recent automated user interface generation tools were developed: SUPPLE [11] considers interface generation as an optimization problem, iCrafter [12] uses hand-created templates, XIML [4] relies on user-specified size constraints in choosing appropriate widgets, and TIMM [8] is a system for automatic selection of interactors to accommodate reduction in screen size.

Though much has been developed for both automated user-interface generation and adaptive visualization, some applications need a system that can both adaptively visualize and generate user interfaces for a variety of display devices. In the following sections, we will give an overview of the PAVIS architecture, discuss the components in detail, and finally describe some prototype results.

3. Architecture

PAVIS (Pervasive Adaptive Visualization and Interpretation Service) is based on a client/server model. The core architecture (Figure 1) is composed of four components: the Visualization and User Interface Engine, Data Filtering Service, Interface Generation Service and Scalable Rendering Service. This gathering of services decouples the visual- and user-interface creation from the display device so that thin-clients (clients with minimal computing capacity) can be handled.

The Visualization and User-Interface Engine

This module is responsible for coordinating between the data source, information about the capabilities of a device (PDA, tiled display etc) and a specification of rules for handling the salience of information. The visualization and user-interface engine retrieves the raw data and related information from the Pantheon gateway server (Figure 1). The related information includes structure of the data, relative salience of the data attributes and the data-specific visualization rules. Information about the display device, such as screen resolution and networking capabilities are obtained from the devices. These data- and device-specific rules are efficiently merged and filtered to obtain a new set of rules that act as an input to the data-filtering and user-interface generation service. As an example from our scenario, the data rule set could specify priorities to data attributes such as: alerts have higher priority than events and based on the device information for a cell phone, a new filtered rule is dynamically generated which specifies “display alerts only” and updates user-interface related rules to include only relevant widgets.

Figure 2 - PDA user visualizing real-time change in traffic congestion.

Figure 3 - 100-Megapixel tiled display depicting real time traffic alerts (white circles)

The Data Filtering Service

This module performs data filtering on the raw data based on the filtered rules obtained from the visualization and user interface rule engine. In our scenario, if there are several alerts occurring around a small radius, displaying all the alerts on a cell phone makes it un-interpretable. A possible alternative is to mark the whole region of that radius as an area of importance and as the user zoom level changes, display the individual alerts based on the available display area. Moreover, separate color schemes to identify the alerts with different priority could be applied. To achieve this kind of interactive and adaptive behavior, efficient data filtering must be applied based on several constraints such as display area, user's area of interest, and data-specific attributes.

The User Interface Service

The goal is to develop a mechanism for the generation of user interfaces that are suited to a variety of display screens [7], ranging from the powerful tiled displays to the small cellular phones. Each platform has its own constraints: some devices are immobile (e.g., a desktop computer) while others are mobile (e.g., PDA); some support extensive graphical capabilities [6] (e.g., wide-band tiled displays). Several user interfaces need to be developed in order to handle this wide range of clients. The user interface engine has a generic rule set built-in, which is then tailored according to the user's device capabilities based on the filtered rules. Also, an adaptive user-interface based on visualized data context can be generated. For instance, to perform magnification operations, a desktop application could be equipped with a slider, whereas a PDA could use the stylus, and a cell phone could use up/down navigation buttons to obtain the same degree of interactivity. As mentioned in the data filtering section, detailed individual alerts on a magnified level can be made interactive by changing the user interface at run-time to handle mouse events on a regular desktop; on a cellular phone, alerts would be numbered and keypads could be used to interact with individual alerts.

The Scalable Rendering Service

Graphical capabilities of the clients vary widely requiring applications to have the capability to perform scalable rendering. By scalability, we mean the system should be able to render data from few thousand pixels to hundreds of megapixels, or from a few primitives to several hundred of thousands primitives. For instance, a PDA client may need only 280 x 320 pixel resolution, whereas a wide-band display might need an image as large as 4000 x 4000 pixels. The challenge of achieving high-quality and fast rendering outputs can be handled by the use of distributed rendering techniques (e.g., using parallel rendering cluster).

4. Current Status

A proof-of-concept prototype was developed manually. It can handle multiple PDA clients and tiled displays with minimal data filtering techniques and generation of visualizations. The processed sensor data is obtained from the Pantheon Gateway Testbed in the form of flat text files and were filtered using pre-defined rules to indicate change in regions of congestion, volume, speed and occupancy. The same procedure is performed on alert, event and weather data input.

PAVIS is designed around XML-based rule sets that specify the data structure, the relative salience of the data attributes and high-level policies (e.g., data attribute X is more important than Y). The reason for choosing XML, as the primary input to the system, is that it could be dynamically integrated with the SVG based visualization routines and with languages like XIML [4] for automated user interface generation. The current prototype uses Batik's [15] SVG rendering capabilities, and the-user interface components were built using Sun JFC/Swing toolkit [16]. Figures 2 and 3 illustrate users

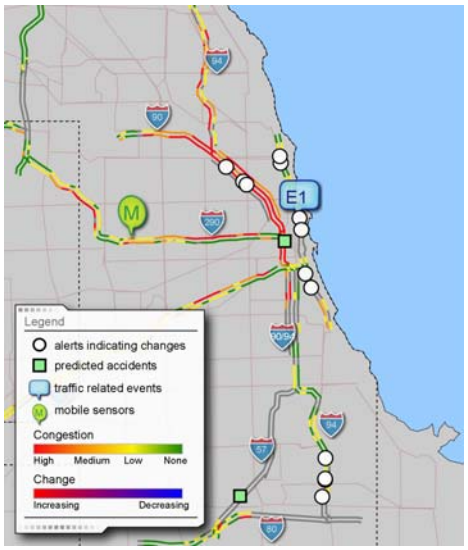

Figure 4 - Real-time traffic congestion map

interacting with both a PDA and a high-resolution tiled display to visualize real-time Chicago traffic conditions. The user interfaces were built by hand using Eclipse user interface editor [19] for the tiled display client, and with IBM Studio Device Developer [18] for the PDA. For the high-resolution tiled display, visualizations were managed using SAGE (Scalable Adaptive Graphics Environment), a graphics streaming architecture supporting high-resolution collaborative scientific visualization [17]. Figure 4 shows the real-time congestion map with several layers enabled (e.g., alerts, events, mobile sensors and predicted accidents). Designing user interfaces and visualizations by hand individually, for a PDA and a tiled display was very time consuming and tedious, hence paving the way for a system such as PAVIS.

5. Conclusion and Future Work

We have presented PAVIS, an approach for performing adaptive visualization and user-interface generation based on device-specific information (e.g., resolution, networking capability). With this system, applications will be able to handle clients irrespective of their working environments and help users to visualize data and interpret meaningful information. Not only display characteristics can be expressed in PAVIS, other domain could be considered such as networking capabilities and power management, which are crucial to mobile devices. Hence, a deeper analysis of rule based visualization and interaction approach is needed to provide a more comprehensive system. This architecture also needs to be applied to a variety of domain specific data sources to prove the versatility of this approach.

Acknowledgements

This work was supported in part by the U.S Army Pantheon Project and National Science Foundation. The Electronic Visualization Laboratory (EVL) at the University of Illinois at Chicago specializes in the design and development of high-resolution visualization and virtual-reality display systems, collaboration software for use on multi-gigabit networks, and advanced networking infrastructure. This material is based upon work supported by the National Science Foundation (NSF), awards CNS-0224306, CNS-0420477, OCI-0229642 and OCI-0441094, as well as the NSF Information Technology Research (ITR) cooperative agreement (OCI-0225642) to the University of California, San Diego (UCSD) for "The OptIPuter." EVL also receives funding from the National Institutes of Health, the State of Illinois, the Office of Naval Research on behalf of the Technology Research, Education, and Commercialization Center (TRECC), and Pacific Interface on behalf of NTT Optical Network Systems Laboratory in Japan. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the funding agencies and companies.

References

[1] B. Rogowitz, L. Trenish, *An architecture for rule based visualization*, Proc of the fourth conference on Visualization'93, San Jose, California, 1993, pp 236-243

[2] L. Jiang, H. Liu, M. Parashar, D. Silver, *Rule-Based Visualization in a Computational Steering Collaboratory*. International Conference on Computational Science 2004: 58-65

[3] S. Lok and S. Feiner, *A survey of automated layout* SmartGraphics Symposium '01, pages 61--68, Mar. 2001.

[4] A. Puerta and J. Eisenstein, *XIML: A Common Representation for Interaction Data.* In Proceedings of the 7th international conference on Intelligent user interfaces, pp. 214-215, 2002.

[5] S. Fiener, J. Mackinlay, and J. Marks, *Automating the design of effective graphics.Tutorial notes.* AAAI '93, Washington DC, July 11-16, 1993.

[6] J. Eisenstein, J. Vanderdonckt, A. Puerta, *Applying Model-Based Techniques to the Development of UIs for Mobile Computers*, Proceedings on Intelligent User Interfaces, Santa Fe, 2001.

[7] M. Satyanarayanan, *Fundamental Challenges in Mobile Computing*, Proc of the fifteenth annual ACM symposium on Principles of distributed computing, ACM Press, New York, 1996, pp-17

[8] J. Eisenstein and A. Puerta, *Adaptation in Automated User Interface Design*, Proc of IUI'2000 (New Orleans, 9-12 January 2000), ACM Press, New York, 2000, pp 74-81

[9] J. Vanderdonckt, L. Bouillon, and N. Souchon, *Flexible Reverse Engineering of Web Pages with VAQUISTA*. Proceedings of the IEEE 8 th Working Conference on Reverse Engineering. Stuttgart, October 2-5, 2001. IEEE Press, pp. 241-248.

[10] R. Grossman, M. Sabala, J. Alimohideen, A. Aanand, J. Chaves, J. Dillenburg, S. Eick, J. Leigh, P. Nelson, M. Papka, D. Rorem, R. Stevens, S. Vejcik, L. Wilkinson, and P. Zhang, *Real Time Change Detection and Alerts from Highway Traffic Data.* ACM/IEEE SC 2005 Conference (SC'05).
[11] S. Ponnekanti, B. Lee, A. Fox, P. Hanrahan, and T. Winograd, *ICrafter: A service framework for ubiquitous computing environments*. Lecture Notes in Computer Science, 2201:56--??, 2001.
[12] K. Gajos, D. Weld, *SUPPLE: Automatically Generating User Interfaces*. In Proceedings of IUI'04. Funchal, Portugal, 2004.
[13] K. Gajos, D. Weld, *Automatically Generating User Interfaces For Ubiquitous Applications.* In Workshop on Ubiquitous Display Environments, Nottingham, UK, 2004.
[14] http://www.w3.org/XML/
[15] http://xml.apache.org/batik/index.html
[16] Java Foundation Classes: Now and the future, Whitepaper, http://java.sun.com/products/jfc/whitepaper.html
[17] B. Jeong, R. Jagodic, L. Renambot, R. Singh, A. Johnson, J. Leigh. *Scalable Graphics Architecture for High Resolution Displays.* Presented at IEEE Information Visualization Workshop 2005, Minneapolis, MN, 10/24/2005
[18] IBM Websphere Studio Device Developer. http://www-306.ibm.com/software/wireless/wsdd
[19] http://www.eclipse.org/