Visualization of Software
Guest Editors: J. Ángel Velázquez-Iturbide and Cristóbal Pareja-Flores

Joint issue with NOVÁTICA and INFORMATIK/INFORMATIQUE

2 Presentation: Representing Abstractions – Cristóbal Pareja-Flores and J. Ángel Velázquez-Iturbide, Guest Editors

4 Software Visualization: An Overview – Luis M. Gómez Henríquez
We present the concept of visualization and its features, and summarize some of its main applications in Computer Science.

8 Program Visualization in Theory and Practice – Mordechai Ben-Ari
Visualization and animation of programs improves understanding by novice students. We survey research programs which show that graphical representations are effective only under certain conditions.

12 A User-Centred Approach for Designing Algorithm Visualizations – Sami Khuri
Algorithm visualizations or the graphic depictions of algorithms in execution are used in explaining, designing, analyzing algorithms, and in debugging, fine-tuning, and documenting programs. This paper gives a brief overview of the requirements analysis.

17 Incorporating Algorithm Visualization into Educational Theory: A Challenge for the Future – Thomas L. Naps
The primary goal of this paper is to examine algorithm visualization and the role it can play in the context of learning models recognized by educational theorists.

22 User Interface Generation: Current Trends – María Dolores Lozano, Pascual González and Isidro Ramos
The more easy the interface is to use, the more difficult is its development. We tackle the subject of user interface generation within the research area of Human-Computer Interaction.

27 Visualization Designs for Constraint Logic Programming – Manuel Carro and Manuel Hermenegildo
We address the design and implementation of visual paradigms for observing the execution of constraint logic programs, aiming at debugging, tuning and optimisation and teaching.

35 Software Visualization Generation Using AnimalScript – Guido Rößling and Bernd Freisleben
Most software visualization tools address specific application areas, forcing users to shift between tools. This article introduces the basics of an algorithm animation tool for arbitrary software visualizations.

41 Automatic Web Publishing of Algorithm Animations – Fernando Naharro-Berrocal, Cristóbal Pareja-Flores, J. Ángel Velázquez-Iturbide, Margarita Martínez-Santamarta
We report on a functional programming environment to generate Web pages containing algorithm animations by means of simple dialogs to specify the graphical style of visualizations, the animation interactions, etc.

Coming issue: “Voice over IP”
Presentation

Representing Abstractions

Cristóbal Pareja-Flores and J. Ángel Velázquez-Iturbide

Introduction

Webster’s dictionary defines to visualize as “to make visible” or “envisage to form a visual mental image”. Therefore, to visualize something is to make it visible by displaying it in one or more media using a graphical language, often enhanced by natural language. A company’s sales chart is an example of a simple visualization of a complex function of that company. “A picture tells a thousand words” and we are constantly reminded of this by the plethora of maps, diagrams, road signs, advertisements, TV, the Web – to name but a few – that enrich our daily lives. It would be very difficult to imagine an engineering process without a visual representation of its intended output. Here, visualization of the goal greatly assists understanding of the analysis, design, construction and use of the product.

All stages in the life cycle of software, from establishment of need, through build and test to deployment and maintenance, are greatly facilitated by visualization. Software has been visualized from its beginnings, although many techniques are not universally accepted. For instance, a flowchart is a graphical static representation of a program and fairly widely accepted; pretty-printing1 program statements is another static, rudimentary form of visualization, but as it is based on the aesthetic criteria of the programmer a more limited audience can benefit.

The continuing quest for better graphical representations has created substantial research on software visualization. Sorting Out Sorting, the video by Ronald Baecker in 1981, containing animations of different sorting algorithms, is commonly considered as a milestone in software visualization. Other key systems include Balsa, developed by Mark Brown in 1984, Tango by John Stasko in 1990 and Pavane by G.-C. Roman et al. in 1992. Simultaneously, the number of methodologies offering visual assistance in design (CASE tools with different kind of diagrams) and programming (visual programming environments) grew substantially. Multimedia and Web technologies have also played a role in this evolution since they facilitate the integration of visualizations with other media, especially textual explanations.

There are associated areas where visualization plays an important role, such as visual programming. But here the purpose is different. Where software visualization uses graphical representations of software as an aid to understanding, visual programming seeks to specify programs more easily by using graphical notations. Think of design diagrams, say, as a kind of high-level visual programming.

Monographic issue on Visualization of software

This monographic issue emphasises the educational use of software visualization. It highlights the current interest in pedagogical design of algorithm animations by the international community that is researching innovations in Computer Science education.

The monograph starts with an overview of software visualization by Luis Gómez, at the Universidad Politécnica de Madrid. His article discusses information visualization in general, and software visualization in particular. Classification, history, problems and uses of software visualization are thoroughly explored.

The next three articles consider the design and use of software visualization, mainly in the field of education. Morchedai Ben-Ari, at the Weizmann Institute of Science, analyses the educational impact of visualizations. He describes past empirical experiments about the effectiveness of learning facilitation using graphical representations in general and software visual-

1. A pretty printer is a program that formats a high-level program source text according to indentation rules based on the syntax and visualizing the program’s structure.

The Guest Editors

Cristóbal Pareja-Flores holds an M.Sc. degree in Mathematics and a Ph.D. in Computer Science. Since 1998, he has worked as an associate professor at the Dept. Sistemas Informáticos y Programación, Universidad Complutense de Madrid. His interests include innovation in CS education, research on functional and concurrent programming, and programming environments. In these fields, he has collaborated in and lead several research projects as well as contributing to both national and international publications. He is the editor of the technical section “Computer Science Education at the University” of Novática.

J. Ángel Velázquez-Iturbide holds an M.Sc. degree (1985) and a Ph.D. (1990) in Computer Science from the Universidad Politécnica de Madrid. He is an associate professor, previously at the Universidad Politécnica de Madrid, and since 1997 at the Universidad Rey Juan Carlos. He is the coordinator of the Computer Languages and Systems area at his university. His interests are programming tools and environments, software visualization, multimedia and Web applications, all of them mainly in connection with seeking innovation in computer science education. He is an associate editor of the “Computer Languages” section of Novática. He is a member of ACM, IEEE Computer Society, AACE and ATI.
ization in particular. He explains these findings from the point of view of the pedagogical theory of constructivism, and also describes several educational experiences in visualization. Sami Khuri, at San Jose State University, points to the problems inherent in the absence of careful planning in the design and use of algorithm animation systems. His article describes the different factors constituting the preliminary phase of requirement analysis, with particular emphasis on the user. Finally, Thomas Naps, at Lawrence University, analyses the role that algorithm visualization can play within four particular learning models.

As already mentioned, software visualization is related to visual programming and design methodologies. In their article, María Dolores Lozano, Pascual González and Isidro Ramos make a new proposal for the design of user interfaces. Here, visual representations (diagrams) are used to design a software product of a visual nature, namely user interfaces.

The issue ends with three articles that describe specific systems for software visualization. Manuel Carro and Manuel Hermenegildo, at the Universidad Politécnica de Madrid, propose several visualizations for data and constraints in logic programs, as implemented in the VIFID and TRIFID tools. Guido Rößling and Bernd Freisleben, at the University of Siegen, describe the main features of the algorithm animation system ANIMAL and its script language ANIMALSCRIPT, as well as its facilities for relative object placement, international support, diagnosis output, and ANIMALSCRIPT extension. Finally, Fernando Naharro, Cristóbal Pareja, Ángel Velázquez and Margarita Martínez describe the facilities of the functional programming environment WinHIPE to automatically generate both visualizations of expressions and Web pages. Every generated page has the typical structure of a lesson on an algorithm and consists of the problem statement, an algorithm description, a functional program and an animation.

### References and recommended reading

References to software visualization abound, mainly in conference proceedings. Apart from the general references we have included in the articles, the reader can find more about specific systems from the following list.

### Bibliography

- C. L. Jeffery, Program Monitoring and Visualization, Springer-Verlag, 1999

### Conferences

- SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education (ITiCSE), last edition: [http://www.cs.ucl.ac.uk/events/iticse2001/](http://www.cs.ucl.ac.uk/events/iticse2001/)
- IEEE Symposium on Visual Languages (VL), last edition: [http://www.computer.org/proceedings/vl/0216/0216toc.htm](http://www.computer.org/proceedings/vl/0216/0216toc.htm)

### Web pages

- Graph Drawing, [http://www.cs.brown.edu/people/rt/gd.html](http://www.cs.brown.edu/people/rt/gd.html)

### The English editors of this issue

Mike Andersson, Richard Butchart, David Cash, Tracey Darch, Laura Davies, Nick Dunn, Rodney Fennemore, Terry Fowler, Hilary M. Green, Roger Harris, Jim Holder, Brian Robson, and Derek Tripp have carefully vetted the papers, improved the style and corrected the grammar.