An Overview of 3D Software Visualization — **Source link**

Alfredo Raúl Teyseyre, Marcelo Campo

Related papers:

- Visualization of the Static Aspects of Software: A Survey

- Software systems as cities: a controlled experiment

- Software Visualization: Visualizing the Structure, Behaviour, and Evolution of Software

- Software visualization in software maintenance, reverse engineering, and re-engineering: a research survey

- Virtual but visible software

Share this paper:

An Overview of 3D Software Visualization

Alfredo Teyseyre and Marcelo Campo, *Member, IEEE,*

Abstract—Software visualization studies techniques and methods for graphically representing different aspects of software. Its main goal is to enhance, simplify and clarify the mental representation a software engineer has of a computer system. During many years, visualization in 2D space has been actively studied, but in the last decade, researchers have begun to explore new 3D representations for visualizing software. In this article, we present an overview of current research in the area, describing several major aspects like: visual representations, interaction issues, evaluation methods and development tools. We also perform a survey of some representative tools to support different tasks, i.e., software maintenance and comprehension, requirements validation and algorithm animation for educational purposes, among others. Finally, we conclude identifying future research directions.

Index Terms—3D software visualization, software comprehension, information visualization, 3D graphics, human-computer interaction.

✦

1 INTRODUCTION

Developing software systems is an arduous task, involving a set of related phases that spawn along the software lifecycle. During all these phases, software engineers need different ways to understand complex software elements. In this context, the use of interactive graphic presentation of data can support significant help to facilitate the analysis and comprehension of such complex information. In fact, experiences in software engineering and visualization areas, confirms that a visual representation of a software system can enhance its understandability and reduce its development costs [1]. Nonetheless, there is a demand for program understanding techniques to graphically represent different aspects of software [2].

The essence of software visualization consists of creating an image of software by means of visual objects. These visual objects may represent, for instance, systems or components, or their run-time behavior. As a result, engineers can obtain an initial perception on how software is structured, understand the software logic, and explain and communicate the development. Effective graphical representations may provide a closer match to the mental model of users than textual representations and take advantage of user's perception capabilities [3–5]. Actually, the human visual system constitutes a massively parallel processor that provides the highest-bandwidth channel into human cognitive centers [4].

Software visualization in 2D space has been extensively studied. Several authors [6–11] put together collections of papers that reflect the evolution and different categories of the area. In addition, a number of taxonomies of software visualization have been proposed [12–16]. However, as a result of hardware advances, many applications nowadays support 3D

graphic capabilities. The inclusion of aesthetically appealing elements, such as 3D graphics and animation, increases the design's appeal, intuitiveness and memorability of a visualization, but also eases perception of the human visual system [17–19].

Although there is a debate on 2D vs. 3D in the information visualization area [20–22], the use of 3D software visualization has the potential to aid in the development process. 3D software visualization may transform the way that knowledge gathering activities take place during software engineering phases [11, 23]. In this context, this article reports on 3D software visualization work, identifying main directions, techniques, problems and evaluation issues in the area. Our goal is to present an overview of the current state of the art, to provide entry points into the literature and to point out challenges that arise when visualizing software in 3D. The remainder of this paper is organized as follows: First, we introduce the Information Visualization and Software Visualization areas, and we address the definition of terms related to these fields. Then, we provide an overview of the current state of research describing several issues such as 3D visual representations, interaction mechanisms, evaluation methods, and development tools. In addition, we survey some representative works in the area. Finally, we describe future research directions.

2 VISUALIZATION

Card et al. [3] define visualization as *"the use of computer-supported, interactive, visual representations of data to amplify cognition"*, where cognition is the acquisition or use of knowledge (see Figure 1). These graphical representations can convey complex ideas with clarity, precision and efficiency [24]. In fact, the human visual system (i.e., the eye and the visual cortex of the brain) constitutes an effective parallel processor that supports the maximal communication channel into human cognitive centers [4]. Furthermore, the visual system frees cognitive

• *The authors are with the ISISTAN Research Institute, UNICEN University, Tandil, Argentina.*
  E-mail: {teyseyre, mcampo}@exa.unicen.edu.ar.
• *Marcelo Campo is also with the CONICET.*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

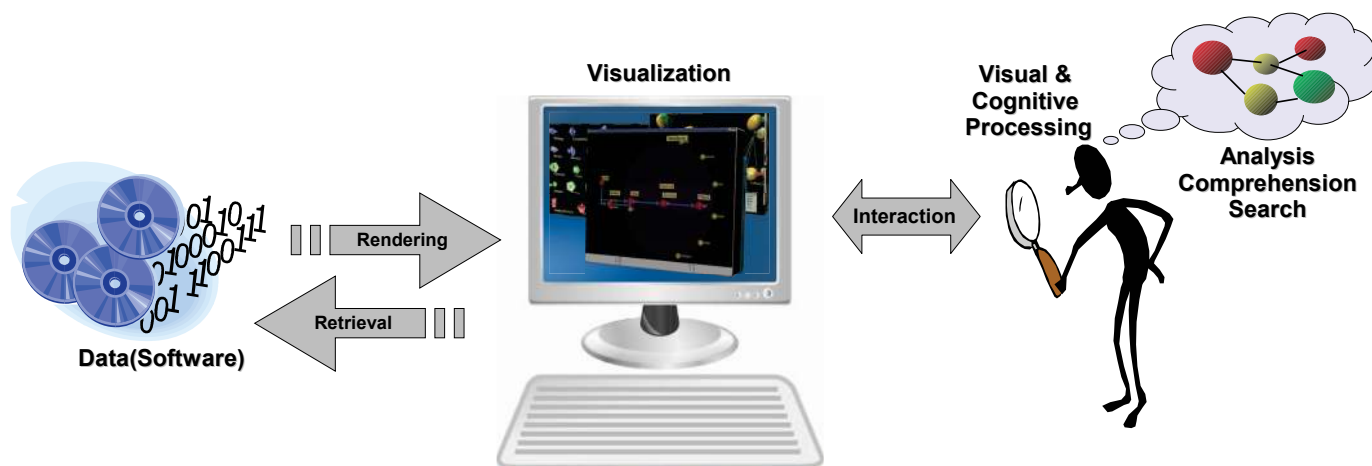IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS

2

Figure 1. Visualization process

capacities by shifting part of processing to it [3]. For example, as a result of visualization, scientists have changed their way of thinking, since they now say they can not do scientific research or communication without visualization [25]. Specially, visualization is a powerful tool that may help users to perform distinct types of cognitive processes [26]:

- *Exploratory*: The user does not know what he is looking for (Discovery).
- *Analytical*: The user knows what he is looking for in the data, trying to determine if it is there (Decision-making).
- *Descriptive*: When the phenomenon represented in the data is known, but the user needs to have a clear visual verification of it (Explanation).

Visualization as a research field is categorized into two major sub-fields: *Scientific Visualization* and *Information Visualization* [3, 4]. Scientific visualization typically represents objects or concepts associated with phenomena from physical world with an inherent spatial component (e.g., chemistry, meteorology, or human body). For example, Figure 2a shows an application (*nanoManipulator*) that provides a virtual-reality interface to a scanned-probe microscope [27]. On the other hand, information visualization typically involves non-spatial data, that is abstract concepts and relationships (e.g., financial data, bibliographic sources or software). For instance, Figure 2b shows a 3D metaphor (*RotaryDiagram*) that helps to visualize information evolution [28]. In particular, Software Visualization is a specialized area of Information Visualization, which focuses on improving software comprehension by providing a tangible representation of abstract software concepts. Also, alternative views of the visualization field have been recently proposed, and they may inspire research ideas in hybrid visualization areas [29].

2.1 3D Visualization

In short, 3D approaches try to create visualizations that are closer to real world metaphors or to improve space

(a) NanoManipulator application: virtual microscope. Image courtesy of the UNC Computer-Integrated Systems for Microscopy and Manipulation NIH 5-P41-RR02170-21
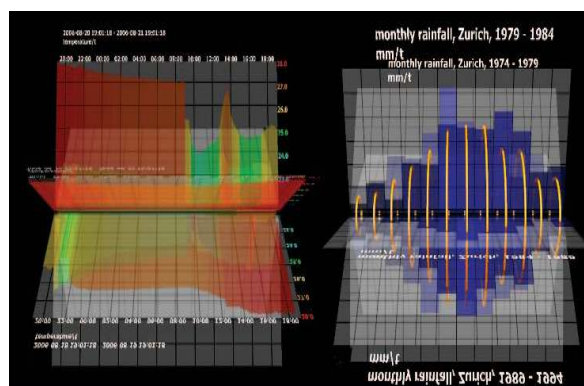
(b) HANNAH Information Visualization Framework: Rotary Diagram and Rotary Diagram with Average Rings. Image courtesy of K. Einsfeld, A. Ebert, and J. Wolle. © 2007 IEEE

Figure 2. Scientific Visualization & Information Visualization examples

usage by adding an extra dimension. The user is able to rotate and move 3D objects and navigate inside a 3D world. Some approaches propose using a 2D layout seen under a 3D perspective with interaction limited to 2D, that is, a 2.5D approach.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS

3

(a) 2D Augmented view: a bubble sort visualization

(b) 2D Adapted view: software release history

(c) Inherent 3D view: Metaballs. Image courtesy of J. Rilling and S. P. Mudur. © 2002 IEEE
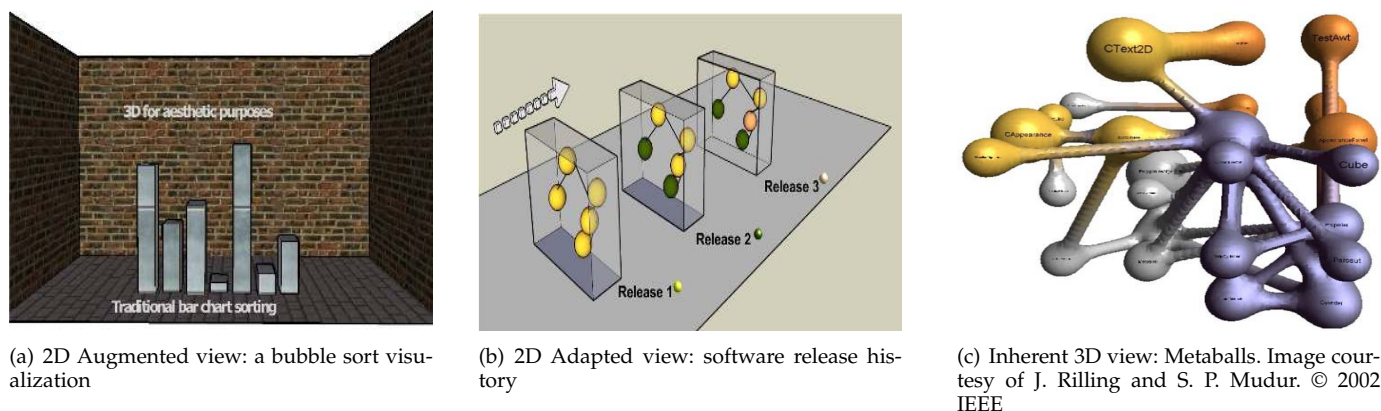
Figure 3. 3D View Categorization

As mentioned in Introduction, there is a controversial debate on 2D vs. 3D in information visualization area. In order to analyze and identify strengths and weaknesses of 3D/2D, we first review a categorization of 3D visualizations that helps in the analysis [20]:

1) **Augmented 2D views**: Typical 2D visualizations where the third dimension is added just for aesthetic purposes. For example, Figure 3a shows a 3D presentation of a traditional 2D bar chart sorting algorithm [30].

2) **Adapted 2D views**: 2D visualizations extended to 3D to encode additional information. To illustrate this, Figure 3b presents a 3D visual representation of a software release history that displays the structure of the system in 2D planes, and uses the third dimension to display historical information [31].

3) **Inherent 3D application domain views**: This category includes computations involving inherent three-dimensional entities. For instance, Figure 3c represents a software system and its relationships using a *Metaball* metaphor, that is, a 3D modeling technique commonly used to represent complex organic shapes and structural relationships in biology and chemistry [32].

In general, the use of the third dimension in Category 3 is out of discussion. Nonetheless, recent research in specific domains shows that 2D and 3D presentations are useful for different task types, and hence combined 2D/3D displays are suggested [33, 34]. On the other hand, the question of the benefits offered by 3D over 2D still remains in the other categories. Several authors [35, 36] state that when 2 dimensions are enough to show information, it is not desirable to add a third dimension. This extra dimension should be only used to visualize a data set semantically richer. However, other authors think that 3D presentations facilitate perception of the human visual system [17, 18]. They believe that the inclusion of aesthetically appealing elements, such as 3D graphics and animation, can greatly increase a design's appeal, intuitiveness and memorability of a visualization [19]. For example, Irani and Ware compared 2D

UML diagrams to *geon* diagrams (3D shaded solids), and they found out that users can identify substructures and relationship types with much lower error rates for *geon* diagrams than for UML diagrams [18]. In addition, the use of 3D presentations provides a greater information density than two-dimensional ones [37]. For example, an experiment [38] suggests that larger graphs can be interpreted if laid out in 3D and displayed with stereo and/or motion depth cues to support spatial perception. Also, this extra dimension helps to have a clear perception of relations between objects by integration of local views with global views [39] and by composition of multiples 2D views in a single 3D view [18, 40]. Lastly, 3D graphics similarity with the real world enables us to represent it in a more natural way. This means that the representation of the objects can be done according to its associated real concept, the interactions can be more powerful (ranging from immersive navigation to different manipulation techniques) and the animations can be even more realistic.

On the other hand, several problems arise, such as intensive computation, more complex implementation than two-dimensional interfaces, and user adaptation and disorientation. The first problem can be addressed using powerful and specialized hardware. Moreover, development complexity is reduced using several tools like 3D toolkits and frameworks [41–46], 3D modeling languages [47, 48], or 3D software visualization frameworks [44, 49–52].

However, one of the main problems of 3D applications is user adaptation. Most users just have experience with classical *WIMP* (windows, icons, menus, pointing) two-dimensional desktop metaphor. Therefore, the interaction with 3D presentations and possibly the use of special devices demand considerable adaptation efforts to these technologies.

Furthermore, it is often difficult for users to understand 3D spaces and perform actions inside them [53, 54]. In particular, as a consequence of a richer set of interactions and more degrees of freedom, users may be disoriented. For example, Plaisant et al. [55] suggest

Strengths	Weaknesses
• Greater information density • Integration of local views with global views • Composition of multiples 2D views in a single 3D view • Facilitates perception of the human visual system • Familiarity, realism and real world representations	• Intensive computation • More complex implementation • User adaptation to 3D metaphors and special devices • More difficult for users to understand 3D spaces and perform actions in it • Occlusion

Table 1
3D strengths and weaknesses

that 3D representations only marginally improve the screen space problem while increasing the complexity of interaction. Moreover, Cockburn and McKenzie [56] evaluated the effectiveness of spatial memory in 2D and 3D, and they found out that navigation in 3D spaces can be difficult and even simple tasks can be problematic. As a way to overcome these limitations 3D enhanced interfaces have been proposed. These interfaces might offer simpler navigation, more compelling functionality, safer movements, and less occlusion than 3D reality [22]. For instance, one alternative to reduce disorientation consists of constraining user navigation with lateral or linear movements [37, 57], or using physical laws such as gravity [58]. Other methods proposed automatic camera assistance during the transition phase from one focus object to the other [59, 60]. In addition, several approaches proposed using landmarks to help users to orient in a 3D world [61–63].

Finally, occlusion may distortion the user's perception of the data-space mainly when the information space is dense [64]. Specially, the occlusion is a serious problem because objects may be occluded and hence appear invisible to the user.

To sum up, there is a vast literature on advantages and disadvantages of 3D vs. 2D with somewhat conflicting results. Table 1 summarizes 3D visualization strengths and weaknesses. Nevertheless, 3D visualizations, if used in ways that exploit their strengths while avoiding their weaknesses [65], may have the potential to aid and improve the development process [11, 23]. In this context, this survey essentially reports results about 3D software visualization.

3 SOFTWARE VISUALIZATION

It is a well-known fact that developing software systems is a complex task that demands developers a number of cognitive tasks, such as, search, comprehension, analysis and design, among others. In this context, software visualization can be a helpful tool to enhance the comprehension of computer programs. In fact, in a recent survey based on questionnaires filled in by 111 researchers from software maintenance, re-engineering

and reverse engineering, 40% found software visualization very necessary for their work and another 42% found it important but not critical [66].

The aim of software visualization is not to create impressive images, but images that evoke user mental images for a better software comprehension [7]. As a result engineers can obtain an initial perception on how software is structured, understand the software logic, and explain and communicate the development. Software visualization combines techniques from different areas like software engineering, data mining, computer graphics, information visualization and human-computer interaction. More precisely, software visualization is a specialized area of information visualization that can be defined as:

"a representation of computer programs, associated documentation and data, that enhances, simplifies and clarifies the mental representation the software engineer has of the operation of a computer system" [1].

Software visualization in 2D has been extensively studied and many techniques for representing software systems have been proposed [6, 67]. However, there exists a demand for effective program understanding techniques and methods [2]. In particular, although the question of the benefits offered by 3D over 2D still remains to be answered, a growing area of research is investigating the application of 3D graphics to software visualization with optimistic results [11, 21, 68]. Researchers try to find out new 3D visual representations to overcome some of the limitations of 2D and exploit 3D richer expressiveness. For example, 3D software visualization has been studied in different areas like algorithm animation for educational purposes [20, 30, 69–71], debugging [72], 3D programming [73], requirements engineering [74–76], software evolution [31, 77, 78], cyber attacks [79], ontology visualization and semantic web [80], mobile objects [81], and visualization for reverse engineering, software maintenance and comprehension at different levels of abstraction (source code [21, 82, 83], object-oriented systems [40, 52, 84–89] and software architectures [23, 90, 91]), among others.

4 VISUAL REPRESENTATIONS FOR 3D SOFTWARE VISUALIZATION

One of the problems that software visualization must address is to find an effective tangible representation of something that has no inherent form [10, 11, 23]. In fact, it is crucial to determine which information to visualize, but also, to define an effective representation to convey the target information to the user and support software engineering tasks [15]. Indeed, the design of a software visualization must address a number of different issues, e.g., what information should be presented, how this should be done, what level of abstraction to support, and so on. For example, a tester wanting to

Figure 4. An example graph representing a software system

find out who was responsible for a bug and when it was injected will look at version history information This information may be visualized as a graph [31] or using an abstract representation based on poly cylinders [78]. Many representations for visualizing software have been proposed. For instance, some visual representations are based on *abstract shapes*, such as graphs [83], trees [44, 92] and geometric shapes [18, 21], and others on *real world* objects, like 3D cities [93–95], solar systems [96, 97], molecules [98], video games [99, 100], metaballs [32], 3D landscapes [52], and social interactions [75], among others.

Therefore, the main challenge is to develop and evaluate effective mappings, from different aspects of the software to graphical representations, in order to provide insight and easier software comprehension [10, 23]. In that sense, Mackinlay proposes two essential criteria to evaluate the mapping of data to a visual representation: *expressiveness* and *effectiveness* [101]. Firstly, expressiveness criteria determine whether a visual representation can express the desired information. Secondly, effectiveness criteria determine whether a visual representation exploits the capabilities of the output medium and the human visual system. Although these criteria were discussed in a 2D graphics context, they can be extended to 3D software visualization. Other researchers discuss desirable properties of visual representations for effective 3D software visualization [23]. These properties may be useful to create new visualizations or to evaluate existing ones.

4.1 Abstract Visual Representations

In this section, we describe several 3D abstract visual representations based on graphs, trees and geometrical shapes. We also analyze their strengths and weaknesses.

4.1.1 Graphs

Many software visualization techniques are based on the graph representation. In short, a graph is a network of nodes and arcs, where the nodes represent entities, such as procedures, objects, classes, or subsystems, while the arcs represent relationships between entities, such as inheritance or method calls. To illustrate, Figure 4 shows a graph view of a software system that we developed and integrated in a 3D Desktop (*Project Looking Glass* by *Sun Microsystems*). In particular, for a review on the state of the art in graph visualization in general see [102], graph visualization in information visualization see [103] and combined approaches see [104].

Graph visualization in 2D space, representing software components around simple boxes and lines, has been applied. However, visualization may result completely incomprehensible, not only as software project complexity increases, but also when visualizing multiples attributes of software, even for small projects [11, 68]. This is a consequence of trying to fit large amounts of information into a reduced space. On the other hand, several authors think that larger graph structures can be viewed in 3D [38, 68]. As we mentioned earlier, an empirical study [38] that measured path-tracing ability in 3D graphs suggested that the amount of information that can be displayed in 3D, with stereoscopic and motion depth cues, exceeds by a factor of three a 2D presentation. Moreover, a re-evaluation of the experiment with new display technologies confirmed the previous experiment and showed a much greater benefit than previous studies [105, 106]. However, other authors think that the most successful network visualizations are small ones (e.g., networks with 10-50 nodes and 20-100 links), where users can count the number of nodes and links, and follow each link from source to destination [107–109]. For instance, Shneiderman and Aris proposed a 2D approach using a layout strategy based on user-defined semantic substrates (non-overlapping regions in which node placement is based on attributes) and interaction capabilities to control link visibility [107]. Dispite the debate, 3D graphs have been applied in different areas of software visualization, such as software configuration management [77], software architectures [90, 91], and object oriented software [18, 83, 84, 110], among many others. However, several issues must be addressed in order to produce effective 3D visualizations:

- *Layout algorithms*: Graphical layout algorithms try to produce well-organized layouts based on graph properties. These layouts, just as in 2D, are crucial for producing comprehensible 3D visualizations. Although 2D classical layout algorithms can be generalized to 3D, layout of 3D graphs may actually differ. For example, aesthetic constraints, such minimization of edge crossing are less important in 3D because arcs are less likely to intersect [68]. Furthermore, the layout of 3D graphs may be more complex as a consequence of trying to address 3D problems such as occlusions, perspective distortions and so on [104]. In particular, several layouts algorithms do a 2D layout and then extend the graph into 3D using some attributes of the nodes [111] or the third dimension for representing time [40]. For
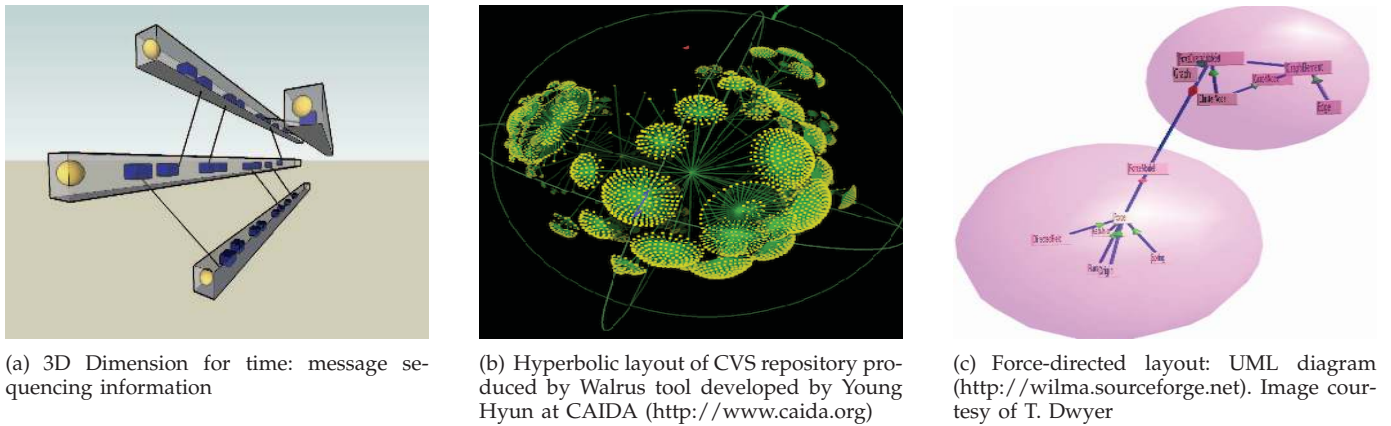
(a) 3D Dimension for time: message sequencing information

(b) Hyperbolic layout of CVS repository produced by Walrus tool developed by Young Hyun at CAIDA (http://www.caida.org)

(c) Force-directed layout: UML diagram (http://wilma.sourceforge.net). Image courtesy of T. Dwyer

Figure 5. Graph layout algorithms

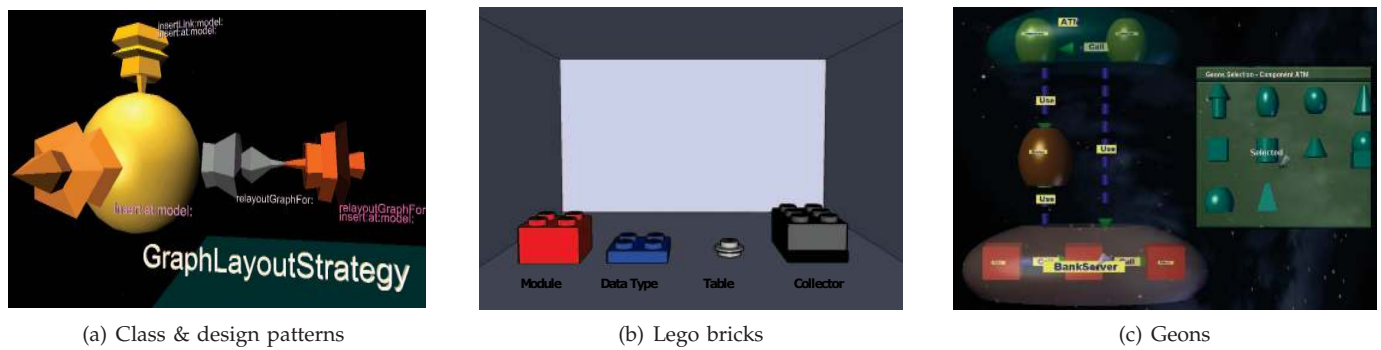(a) Class & design patterns

(b) Lego bricks

(c) Geons

Figure 6. Node representations

instance, Figure 5a shows a 2D layout of modules in a message passing system, where the third dimension displays the message sequencing information. Other techniques compose simultaneously several 2D visualization using orthogonal axes [40]. In addition, 3D hyperbolic layouts [112] have been applied. This kind of layouts provides a distorted view of graph, which makes it possible to interact with potentially large graphs, as shown in Figure 5b. Finally, other layout algorithms use the 3D space without preserving a 2D view from some perspective. For example, Force-Directed Methods [102], described in dimension independent terms, can be applied to 3D [113]. These methods model nodes and edges of a graph as physical bodies tied to springs. These bodies have forces acting on or between them, such as magnetic repulsion or gravitational attraction. For instance, Figure 5c shows a 3D UML Class diagram produced using a force-directed layout [114].

- *Node and link representations*: Different colors and shapes may be used to represent several kind of entities, relationships and software metrics. In general, most common representations include spheres for nodes and cylinders for links. Also, more complex representations may help to visualize additional information. To illustrate, Figure 6a presents a class visualized as volume composed of three semi-axes,

where each axis, represents a design pattern category [115], that is, behavioral, creational and structural, and each pattern is represented by a distinctive polyhedral shape [116]. Other representations use arbitrarily shaped and colored 3D objects such as *Lego bricks* (Figure 6b) [90] or 3D primitives called geons (Figure 6c), which take advantage of human ability to remember and distinguish 3D shapes [18].

- *Clustering*: Although 3D enables us to represent graphs with more nodes and links than 2D, when the number of nodes increases noticeable, the scale problem still remains. In order to address this problem, clustering algorithms have been proposed [68]. In short, clustering helps to visualize large graphs by grouping related nodes. For example, the hierarchical composition of software (i.e., subsystem, module, and file) can help to represent large nested graphs of software systems [68, 114].

4.1.2 Trees

A tree can represent many software entities, such as subsystems, modules or classes, and relationships between them, such as inheritance or composition. Moreover, since trees have no cycles, unlike graphs, they are generally easy to layout and interpret [3]. In general, tree visualization techniques encode hierarchical information using *node-link* diagrams, that is explicit relations,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS

7

(a) Cone Tree. Image courtesy of G. G. Robertson, J. D. Mackinlay, and S. K. Card. © Xerox PARC, Inc.

(b) Hierarchical Net 3D. Image courtesy of M. Balzer and O. Deussen. © 2004 IEEE

(c) Using Circle Data Packing to represent software structure
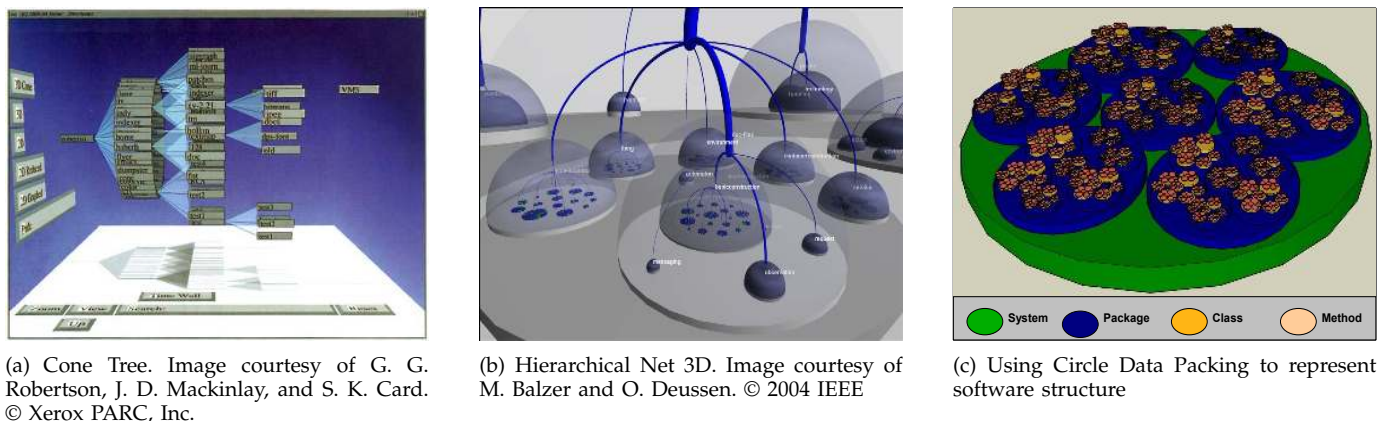
Figure 7. Different tree representations

but also *containment* representations, i.e., relations using space-filling methods. For instance, Figure 7a shows a node-link representation [117], Figure 7c a containment representation [92] and Figure 7b a mixed approach [85].

A well-known node-link representation is *Cone Tree* [117, 118]. This visualization technique for displaying hierarchical information in 3D can show more information than 2D counterpart. The aim of this technique is to allow a greater amount of information to be navigated and displayed in an intuitive manner. Each sub-tree is laid out as a cone with its root at the top of the cone and the children along the cone base. In addition, nodes are semi-transparent rectangles, and when one has been selected, the tree rotates bringing the nodes on the path to the currently chosen node closest to the user. These nodes are also highlighted. Therefore, the animation and interactive selection enable to create a focal point on the structure and shift some of the cognitive load of comprehending the structure to the human perceptual system. However, one of the main problems of this technique is that some of the nodes are occluded, so it is mainly effective for comprehending the overall structure of the tree.

An alternative to visualize trees, is containment or enclosure. Unlike node link representations, containment fills the space. In particular, 3D enclosure tree representations, such as *Information Cube* [92], *Circle Data Packing* [119] and *Beamtrees* [120], are loosely based upon 2D tree-map visualizations [121]. This kind of representation is effective for showing quantitative variables such as code metrics. For example, *Information Cube* is a technique to visualize hierarchical information using nested translucent cubes. Due to this transparency, the user is allowed to view the contents of the cubes and their children, while hiding inner information gradually. Meanwhile, *Circle Data Packing* and *BeamTrees*, overlap nodes to indicate a parent-child relationship resulting more effective than nested representations for the extraction of global hierarchical information (see Figure 7c).

Finally, other approach called *Hierarchical Net 3D* [85] mixes containment and explicit relationships to show the

static structure of object-oriented systems. The hierarchy of packages, classes, methods and attributes is represented using nested hemispheres for packages, circles for classes, and boxes for methods and attributes. Also, relations between entities, such as dependencies between classes, are represented by explicit connections. Furthermore, the visual complexity is reduced by adjusting the transparency of object surfaces to the distance of the viewpoint.

4.1.3 Abstract Geometrical Shapes

Many software visualization tools use traditional node-link diagrams, but sometimes they present scalability or layout problems. In an effort to explore new representations beyond graphs, several visualization techniques were proposed using abstract three-dimensional geometrical shapes [21, 23, 89, 122].

For example, *sv3D* [21, 82] is a tool that supports the visualization of large-scale software to assist in comprehension and analysis tasks associated with maintenance and re-engineering. It is based on *SeeSoft* [123] and *3D File Maps* [111] techniques. As a result of using the third dimension, texture and abstraction mechanism, this tool can represent higher dimensional data than previous 2D views. To illustrate, Figure 8a shows a container that represents a source file, where each poly cylinder represents a line of text from the source code associated
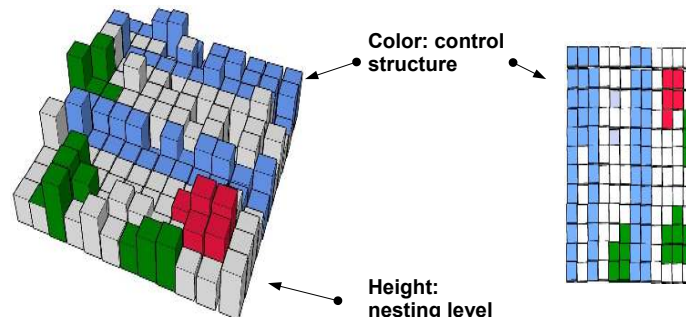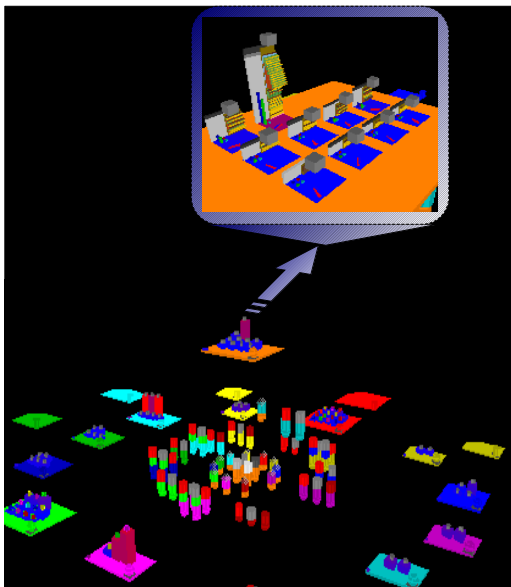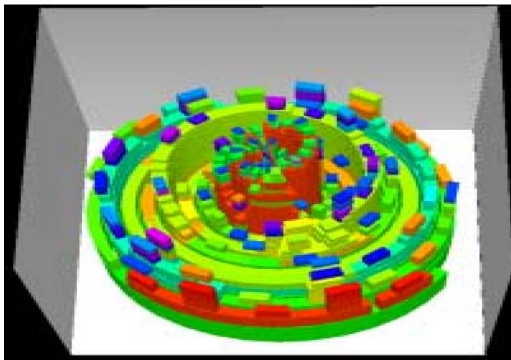

Figure 8. sv3D representation vs. 2D SeeSoft

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS

8

(a) Callstack and FileVis. Adapted from images courtesy of P. Young and M. Munro. © 1998 IEEE

(b) 3D Spiral Stack. Bloom Visualization System (http://www.cs.brown.edu). Image courtesy of S. P. Reiss.

Figure 9. Abstract software representations

with the container. In addition, color is used to represent control structure and height to represent nesting level. On the other hand, a 2D *Seesoft* representation (Figure 8b) of the same file can just represent nesting level using colors.

FileVis [23] is another visualization tool that shows code files represented individually as floating platforms around a central point, which represents the connectivity of the source code files (see Figure 9a). This system not only shows structural information, but also run-time information using a technique called *Callstax* to represent the calling structure of C code with colored stacks of blocks [124]. Similarly, another tool shows full stack information of a trace using a 3D spiral representation to encode time series data in a compact and space efficient manner [122]. For instance, Figure 9b shows a sample trace, where color indicates the routine being called, height indicates stack depth, and width indicates total run time.

4.2 Real World

Trying to find suitable representations of software, several researchers proposed using real world metaphors. These techniques use well-understood elements of the world to provide insights about software. For example, some of these techniques are based on a *City* abstraction [93–95, 125, 126]. In particular, *CodeCity* [126] represents classes as buildings located in city districts, which in turn, represent packages (see Figure 10a). Another visualization [52] that helps to gain an overview of a software system is presented in Figure 10b. It is based on a 3D landscape technique called *ThemeScapes* [127]. This visualization shows the relation "is called by" as 3D landscape. Classes are distributed on the plane and their relative distance is determined by their dependencies. The shaded image and height plot encode the frequency of calls. Two hot spots, corresponding to the two peaks in the image, reveal the most called classes in the system (String and ListIter). Moreover, there exist several other systems based on real world metaphors, such as *ScenarioML*, a requirements engineering tool for validating use cases using social interactions [75], and *Metaballs*, a 3D modeling technique commonly used to represent complex organic shapes and structural relationships in biology and chemistry [32], among others.
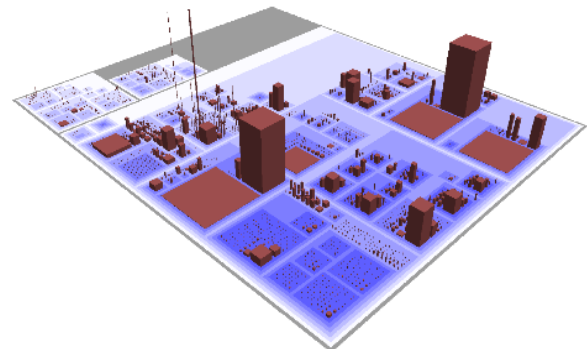
(a) ArgoUML as 3D city. Image courtesy of R. Wettel and M. Lanza. © 2007 IEEE
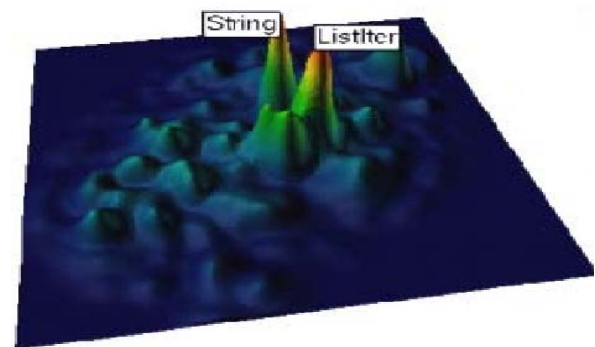
(b) 3D Landscape. Image courtesy of A. Telea and L. Voinea. © 2004 IEEE.

Figure 10. Real world metaphors

5 INTERACTION

Software visualization tools not only have to provide effective visual representations, but also effective interactions styles to ease exploration and help software engineers to achieve insight. However, interacting with 3D worlds is more complex than with 2D *WIMP* interfaces (windows, icons, menus, pointing) [128]. In fact, applications using 3D graphics are essentially different from classical 2D applications and present many challenges. For example, many tasks in 3D applications require the user to manipulate object position and also orientation involving actually six degrees of freedom. Nevertheless, most users have only 2D input devices that should be mapped to a 3D environment. In addition, specialized methods are needed to navigate in a 3D space, due to the increased degree of freedom [60].

In order to perform tasks, interaction techniques provide means for translating user actions into system actions. These interaction techniques can be classified into several categories [54]:

- *Direct Manipulation*: Interaction techniques for manipulating objects provide means to select, position, and rotate objects. Instead of controlling objects through menus or dialogs, users can operate on them. For instance, some systems provide a 3D *manipulator* or *handle* [64, 82], that is, an interactive 3D object that helps to edit and operate on another object. Other systems provide a *virtual hand*, a typical approach used in immersive virtual environments, which is in turn intuitive as simulates a real world interaction [54, 83].
- *User Navigation*: A film can be used to derive a basic metaphor to build 3D dimensional animated graphics: the 3D geometric objects as the actors, the display as the scenario, the lights and the camera [46]. Specially, the notion of *camera* helps to observe the scene, focusing the attention on a part of it from a particular position. Also, the camera zoom can be adjusted like the lens of a real camera. Therefore, the user can manipulate (zoom, pan, rotate) the camera interactively to navigate through the scene. To be more precise, navigation is often the primary task in 3D worlds and refers to the activity of moving through them. In this way, navigation provides means to explore and view information from different perspectives and with different degrees of detail. Some systems enable users unconstrained navigation through the information space [83, 85, 86], however simple 6 degree-of-freedom camera movement through a 3D scene may not be effective in these structures [129]. Others restrict movements in order to lighten possible user disorientation [82, 117]. As an illustration, *Cone Tree* [117] employs a cascading rotation of the 3D cones to bring the desired child nodes to the front. In *sv3d* [21, 82] the visualization can be panned and zoomed in or out, but the position of the camera is fixed. In addition,

several systems provide *semantic zoom*: level of detail changes as user inspects nodes closer. To be specific, as the viewpoint moves towards a particular object, at a certain threshold, the low detailed object representation will be replaced with a high detailed representation [23, 91, 94]. For example, a system may display a class using a box representation at a high level of detail, and as the user gets closer it may represent additional information such as methods, variables, and software metrics.

- *System control*: The user sends commands to an application. As many of the systems use traditional 2D devices, interaction is done using conventional widgets. Also, other techniques include using 3D *widgets* [130, 131], *voice commands* [132] and *hand gestures* [73].

6 EVALUATION

Although research on visualization creates impressive images, every design needs to be tested to determine how useful it is for real people doing real tasks [133–135]. In addition, any usability evaluation of a visualization technique has to address both evaluations of visual representations and interactions styles, because exploration may help to achieve insight that a set of fixed images can not [136, 137]. In the last years, interest in the evaluation of information visualization systems has grown incorporating human-centered principles of interaction and usability [133–135, 138–141].

Most of the methods for usability evaluation were developed for graphical user interfaces and adapted or extended for information visualization. Some well-known methods that have been applied are *analytic* ones (i.e., expert reviews and cognitive walkthroughs) and *empirical evaluations* (i.e., controlled experiments, questionnaires, interviews and focus groups). In particular, see [142] for a detail description of these methods. As a result of the lack of evaluation studies in the area of 3D software visualization this section also reports 2D evaluation studies.

Analytic evaluation involves the analysis of the user interface to discover potential usability problems and guide modifications during the development of the system. For instance, Brown et al. reported valuable feedback in the application of expert reviews during the development of a tool to visualize the execution of parallel algorithms and Stasko et al. [143] evaluated an algorithm animation system using cognitive walkthroughs. Furthermore, several researchers performed theorical analytic studies. For example, Sun and Wong [144] evaluated two commercial UML tools using fourteen criteria based on gestalt-theorical principles. Similarly Blackwell et al. [145] analyzed and compared, using a theoretical framework (Cognitive Dimensions Framework), software engineering notations. A number of taxonomies of software visualization have been proposed as well [12–16, 146], which in turn provide support for qualitatively evaluation and comparison of software visualizations.

On the other hand, empirical evaluation methods collect usability data by observing or measuring activities of end users interacting with a prototype or an actual implementation of the system. To illustrate, Purchase et al. [147] reported an empirical study of variations in UML notation, Lange and Chaudron [148] evaluated new views to support comprehension of UML models, and Yusuf et al. [149] and Guehénéuc [150] assessed how people comprehend UML class diagrams using eye-tracking equipment. As mentioned before, Irani and Ware [18, 151] compared 2D UML diagrams to diagrams using 3D primitives called *geons* and they found out that users can identify substructures and relationship types with much lower error rates for geon diagrams than for UML diagrams. Also, Marcus et al. [152] conducted a usability study to improve a 3D visualization system called *sv3D*. Despite the overall positive impression that users have of this system, this study reveals their inherent difficulty to adapt to a new technology (3D). Storey et al. [153] evaluated the usability of three user interfaces of a reverse engineering tool by observing users completing a set of software maintenance tasks, followed by a questionnaire and an interview. Also, Jones and Harrold [154] evaluated and compared a technique for fault localization (*Tarantula*) with others techniques. In addition, Alwis et al. [155] compared three specialized software exploration tools. Finally, several surveys on software visualization based on questionnaires are reported [66, 156].

Indeed, there is a lack of evaluation studies in software visualization research [7] and specially in 3D software visualization. In fact, it is necessary to conduct user studies in order to gain insight, e.g., under what types of tasks and conditions a particular technique is effective. This knowledge is critical because different analytic tasks require different visualization techniques [135]. Furthermore, many of the empirical studies generally include only simple tasks [139]. Additionally, researchers had begun to identify new areas to improve focusing on obstacles that difficult higher-level analytic tasks [157], and new ways to measure and evaluate visualizations based on insight [158].

7 DEVELOPMENT TOOLS

Information and software visualization applications are difficult to build, requiring mathematical and programming skills to implement complex layout algorithms and dynamic graphics [51]. Moreover, the development of 3D graphics applications is a hard work that consumes much more time than conventional 2D graphics applications, requiring specific knowledge about 3D geometry operations as well [41, 46]. Although 3D graphics libraries, such as *OpenGL* [159], provide a complete application programmer interface (API), they demand significant effort to create even simple visualizations. In particular, most of these APIs require that the application developer cultivate a thorough understanding of a complex programming model, which includes matrix operations, 3D geometry and lighting models [46].

For that reason, several toolkits and frameworks were developed to alleviate these problems. In fact, object-oriented frameworks are a powerful technique to build applications that increases the quality of the software and productivity. In short, a framework represents a reusable design for a specific software domain. To be more precise, using a framework implies the reuse of all the control structure codified in its classes and the production of only the specific code that will be called by the framework. Hence, different applications can be obtained by reusing code and the general design of the application [160]. For example, several frameworks to build 3D graphics applications have been developed: *GRAMS* [43], *GROOP* [46], *IRIS Inventor* [41], and *Java3D* [42], among others. These frameworks, undoubtedly, make the development of 3D graphics applications easier. Similarly, specific frameworks and toolkits for virtual environments (VE) or distributed virtual environments (DVE) have been also proposed: VR *DIVERSE* [161], *Juggler* [162], and a service and component-based framework for DVE [163], among others. In addition, other approaches to develop 3D graphics include 3D modeling languages such as *VRML* [48] and its successor *X3D* [47]. These technologies provide a way to model 3D scenes using a description language. In particular, *X3D* promises to be more successful, due to flexible XML-encoding, modularization through profiles and smarter 3D browsers. Moreover, on the top of *X3D* and XML, *Contigra* [45] introduced a component-based approach to construct interactive, three-dimensional applications, either stand-alone or web-based.

Specially, in the area of information visualization, there also exist several toolkits and frameworks. Some of them address just 2D visualizations [51, 164], but others provide 3D graphics capabilities as well [28, 165, 166]. Moreover, software visualization frameworks have been also developed. In general, these frameworks support different metaphors, several layout algorithms, and binding functions. For instance, *Mondrian* [167] and *Evolve* [168] display 2D visualizations, *Effects* renders UML diagrams in three dimensions [169], *Luthier* [170] helps to visualize object-oriented applications and frameworks, and *BLOOM* [122] provides support for Box trees, File maps, different graph layouts and Point maps. Additionally, *VOGUE* [40] concentrates on integrating a number of 2D views to create a more powerful 3D visualization. This framework has been applied in a number of different visualization scenarios: parallel Linda programs, version control and module management, and a C++ class library browser. Other framework, *SoftVision* [52], supports interactive visualization (2D / 3D) and exploration of the structure, properties, and behavior of component-based systems. Using this framework, developers can freely specify both, the component data to be examined (i.e., what they want to visualize) and how they want to view the data, by assembling pre-packaged components

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS

11

such as data editing, filtering, rendering, and user actions. Finally, *vizz3D* [49, 171] is a framework for creating new software visualizations that enables to configure online views and their mappings instead of hand-coding them.

Researchers also have begun to explore new ways and models to create software visualizations. For example, Bull et al. [172] proposed a model driven approach to assist in the creation of highly customizable interfaces for software visualization. Also, Itoh and Tanaka [173] proposed a component-based framework for generating simple 3D applications that aid users in the use of Web services without the need for programming. The framework uses the 3D media system *IntelligentBox* as its platform [174]. Moreover, an aspect that is gaining attention, is the integration of visualization tools into development environments (e.g., *Eclipse*) that developers use every day [169, 175, 176].

8 3D SOFTWARE VISUALIZATION PROJECTS

This section reviews several 3D software visualization systems. It reports on some representative tools for doing different tasks, e.g., algorithm animation, software configuration management, software maintenance and comprehension, and requirements validation, among others. In addition, Table 2 summarizes the main characteristics of the reported tools, based on categories defined by several software taxonomies [14, 15, 146]. These categories include: *intent* (the purpose of the tool / task supported), *audience* (who will use the visualization), *information* (what aspects of the software are to be represented), *presentation* (characteristics of the output of the visualization), *interaction* (how the user interacts with visualization), *effectiveness* (evaluation of the tool). Other relevant information such as associated *development tool* and *year of creation* is also reported.

8.1 Algorithm Animation

Algorithm animation visualizes the behavior of an algorithm (data and operations). In fact, animating an algorithm has proven to be useful for education and for research in the design of algorithms [178, 184]. For example, well-known algorithm animation systems such as *Polka* [20] and *Zeus* [71] were extended with 3D graphics. Also, *JCAT* system extends previous work (*Zeus*) and provides support for web-based algorithm animation in 3D [178]. In particular, *GASP*, a domain specific animation system, explored 3D animations of computational geometry algorithms [177]. As a consequence of restricting the domain, this system requires much less effort to build animations. To illustrate, Figure 11 shows the animation of a geometric algorithm.

As a result of using the third dimension, it is possible to capture a history of execution (i.e., elementary sorting using classical sticks view extended in z-axis to represent progress), to integrate multiple 2D views to reduce cognitive load (i.e., heapsort 3D view that
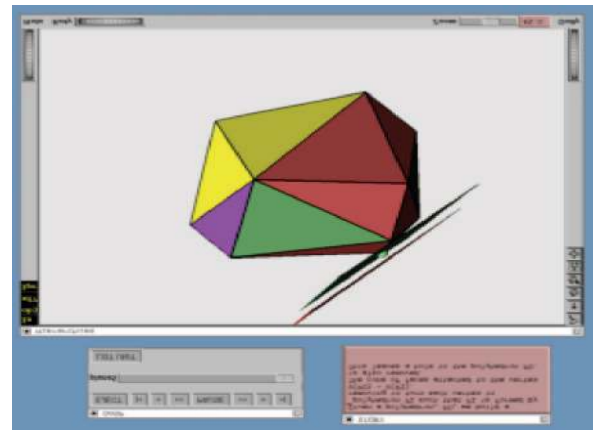

Figure 11. Gasp. Image courtesy of A. Tal and D. Dobkin. © 1995 IEEE

combines a tree view and a sticks view), to display additional information (i.e., shortest-path animation where the third dimension provides state information about the algorithm as it operates on a data structure represented in 2D), and to visualize inherent 3D domains (i.e., 3D geometric algorithms) [20, 177, 185].

8.2 Software Evolution

Usually a software system changes over time because new services are added, bugs are removed, or the technologies get obsolete. In order to make informed decisions about future developments, it is essential to quickly grasp a comprehensive view of the system evolution [31]. For instance, *VRCS* [77] helps to do version control and module management using 3D graphics. Each version history is displayed as a 2D tree by taking the z-axis as time. Files (trees) in the same module are placed physically near when looked along z-axis. In addition, module structure and version history are integrated in one 3D visualization. Also, a certain release of the system is represented as a sphere, and versions
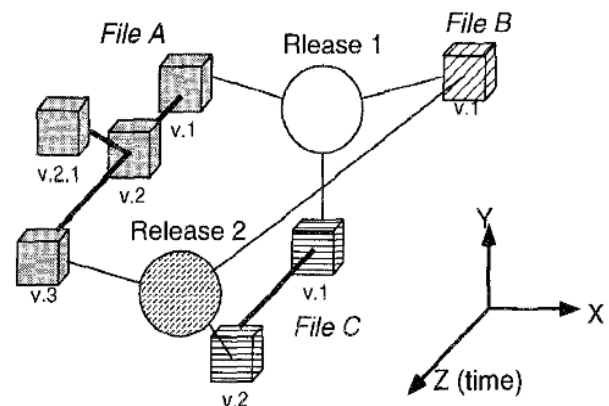

Figure 12. 3D VRCS representation. Image courtesy of H. Koike and H. Chu. © 1997 IEEE

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS

12

System	Audience	Information	Presentation	Interaction	Effectiveness	Development	Year
Algorithm Animation							
Polka [20]	Designers Students	Algorithms, i.e., sort and graph	2D adapted views	Position and direction of viewpoint	Several examples Student courses	Own Toolkit in C++	1993
Zeus [71]	Designers Students	Algorithms, i.e., sort and graph	2D adapted views	Zoom, pan, rotate	Several examples Student courses	PexLib	1993
Gasp [177]	Designers Students	Geometric algorithms	3D world	VCR metaphor Zoom and rotate	Several examples Student courses	Open Inventor	1995
JCAT [178]	Designers Students	Algorithms, i.e., sort and graph	2D adapted views Web-based	Zoom, pan, rotate	Several examples Student courses	Java3D	2001
Software Evolution							
VRCS [77]	Developers Maintainers Re-engineer	Versions & relationships between file versions	3D integrated view of module structure and version history	Select versions and files for check in /out	Comparative experiments reveal faster use than rcs [179]	3D SV framework (VOGUE)	1997
Release History [31]	Developers Maintainers Re-engineer	System structure evolution	3D graphs - z axis time	Zoom, pan, rotate	Case study (13 MLOC)	VRML	1999
EPOsee [180]	Developers Maintainers Re-engineer	dependencies between software components	Partially 3D 3D bar chart	Fixed camera	Several examples (i.e. Mozilla)	Java	2005
Software maintenance and comprehension							
Code level							
sv3d [21]	Developers Maintainers Re-engineer	Source code and related attributes.	Abstract shapes Metrics mapped to visual attributes	Zoom, pan, rotate 3D handler	Several examples Usability studies [152]	3D framework (OpenInventor)	1997
Detailed design							
FileVis [23]	Maintainers Re-engineer	C Source Code	Abstract Shapes. 3D Structure overview, metrics and control	Zoom, pan, rotate. Adaptable Level of detail. VR environment	Guidelines for designing software visualizations	VR toolkit	1998
Code Mapping [83]	Developers	C/C++ source files.	3D Graph. Immersive environment	Zoom, pan, rotate. Virtual hand	Informal field use	VR toolkit	2004
Trace Crawler [181]	Developers Maintainers Re-engineer	classes and inheritance object creations and message sends	3 D Graph. Structural and behavior integrated view	Zoom, pan, rotate Details on-demand.	Two case studies (700 classes and 6000 events)	3D framework (Jun)	2005
Imsovision [110]	Developers Maintainers	Classes and inheritance relationships	Abstract shapes / Graph	VR environment	Analyzing the seven high level tasks of IV	VRML	2001
Unified City [125]	Managers Developers Maintainers	C/C++ / additional software analysis	City metaphor unified single view	Zoom, pan, rotate	Informal use and few examples	3D SV framework (Vizz3D)	2007
Geons3D [18]	Developers Maintainers	UML Class Diagram	Nodes and links are replaced with 3D primitives (geons)	Work focuses just on graphical representations	Experiments show lower error than UML diagrams.	OpenGL	2003
X3D-UML [84]	Developers Maintainers	UML Class Diagram	3D graph	Zoom, pan, rotate.	Informal use and few examples	X3D	2005
Architectural level							
ArchView [90]	Software architect Re-engineer	Software Architecture. Call-tree / part-of from source	Graph and Lego Bricks	Zoom, pan, rotate.	Small example application	VRML	1998
Abacus [91]	Software architect Re-engineer	Sub-system and transaction view, and metrics	Graph Metrics mapped to visual attributes	Zoom, pan , rotate Adaptable Level of detail	A case study (300k LOC)	Not documented	2002
Requirements Validation							
ScenarioML [75]	Stakeholders (even non-experts)	Scenarios and use cases.	Social interactions of agents in 3D	Fixed Camera	Exploratory study	OpenGL	2006
ReqViz3D [74]	Stakeholders (even non-experts)	Z Specifications	3D World Scripting language	Zoom, pan, rotate	Informal use and few examples	3D Framework (Java3D)	2005
PNVis [182]	Stakeholders (even non-experts)	Petri nets	3D World	Zoom, pan, rotate	Informal use and few examples	Java 3D	2004
Other purposes							
Starmine [79]	System Administrators	Cyber attacks	3D multiple metaphor that integrates different views	Fixed Camera	Monitor university/enterprise networks in Japan.	3D Framework (Java3D)	2006
Ontosphere [183]	Developers	Ontology Web services	Graph	Zoom, pan, rotate	Few examples	Java 3D	2006

Table 2
Visualization Tools Summary

composing the release are connected by links (see Figure 12). Similarly, Gall et al. [31] visualized the structure of a system using 3D graphs and represented the historical information by using the third dimension as time. Another tool (*cv3D*) [78] visualizes information extracted from CVS repositories based on visualization techniques provided by *sv3D* [21, 82]. At last, other tool that partially supports 3D graphics is *EPOsee* [180]. This tool provides a 3D bar chart to find dependencies between software components. The third dimension is used to encode additional information of dependencies.

8.3 Software Maintenance and Comprehension

Many 3D visualization tools help software engineers and developers to understand how software works. These tools focus on different level of abstractions (i.e., source code [21, 82], detailed design [40, 52, 84–88] and software architectures [23, 90, 91]) and on different software views (i.e., software structure and/or its run-time behavior). For example, *sv3D* [21, 82] supports the visualization of large-scale software to assist in comprehension and analysis tasks at code level. This tools represents a line of text from the source code using poly cylinders. Different attributes, such as color, position, height and depth can be associated with software properties and line oriented metrics as well (e.g., color may reflect control structure type and height nesting level).

In addition, many software visualization tools provide an overview of structured or object-oriented systems. For instance, *FileVis* [23] visualizes systems developed in C using an abstract geometric representation, and *Code Mapping* [83] uses a 3D graph representation, where nodes representing software artifacts (procedures, variables, calls or data accesses) are mapped to spheres and relationships between software artifacts to lines. In particular, *TraceCrawler* [86, 181] helps to understand both structure (i.e., classes and inheritance relationships) and behavior of an object-oriented system (i.e., object creations and message sends). This visualization displays the static structure of an application using a graph representation (i.e., classes (nodes) and inheritance relationships (edges)). On the other hand, the application dynamic is represented as object instantiations (boxes) and message sends between objects (connectors). The boxes are displayed as towers of instances on top of their defining classes in the class hierarchy view (see Figure 13). The user can map several metrics to visual attributes of the nodes (i.e., width, length and color). Another system that proposes a unified single visualization, although based on a real world metaphor, is *Unified City* [125]. This unified single-view visualization supports different kinds of tasks: it allows to see and communicate current development, quality, and costs of a software system quickly (Figure 14). Also, other approach investigates how to improve the interpretation of node-links diagrams using three-dimensional shaded elements instead of simple lines and outlines [18].
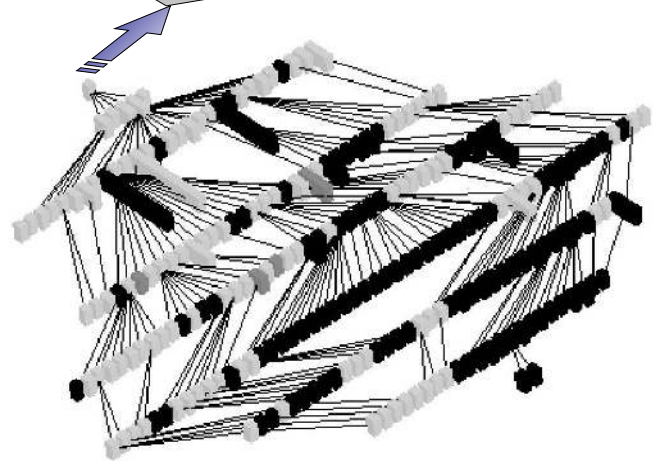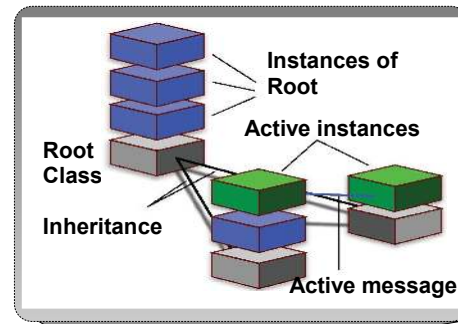

Figure 13. TraceCrawler. Adapted from images courtesy of O. Greevy, M. Lanza, and C. Wysseier. © 2005 IEEE

Finally, tools such as *ArchView* [90] and *ABACUS* [91, 186], focus on architectural abstractions. For example, *ABACUS* is a tool that creates structural abstractions of software architectures as 3D graph layouts for software maintenance, as seen in Figure 15. This visualization shows architectural structures and relationships present in the code, and uses the color (shading) of the spheres and the size of the cylinders to show metrics-based infor-
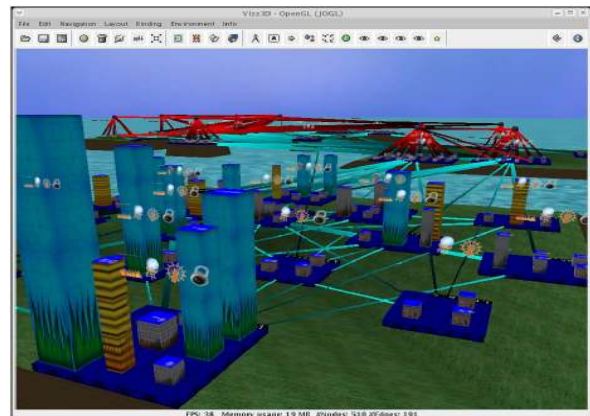

Figure 14. Unified City. Image courtesy of T. Panas, T. Epperly, D. Quinlan, A. Saebjornsen, and R. Vuduc. © 2007 IEEE.
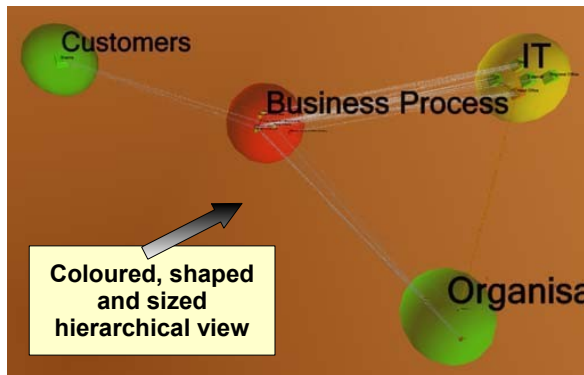
Figure 15. ABACUS: 3D Overall Enterprise Architecture View. The technique is "international patent pending" and has been implemented through the ABACUS product available from Avolution (www.avolution.com.au). Image courtesy of K. Dunsire, T. O'Neill, M. Denford, and J. Leaney. © 2005 IEEE
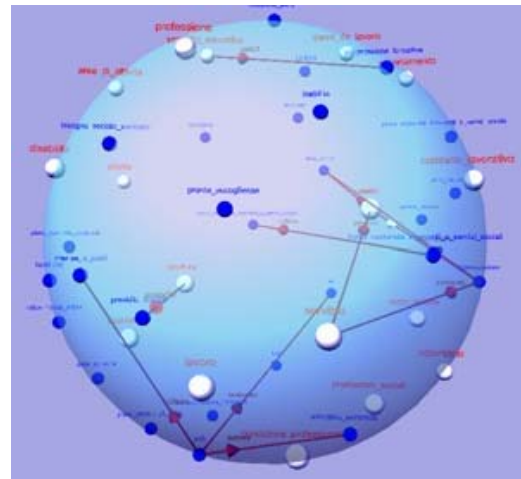

Figure 17. OntoSphere3D: a multidimensional visualization tool for ontologies. Image courtesy of A. Bosca and D. Bonino. © 2006 IEEE

mation (e.g., Lines of Code (LOC), Lines of Comments, Percent Comments, Executable Statements or Complexity).

In conclusion, the third dimension helps to increase information density [82, 86, 125], integrates multiples views (i.e., structure and behavior) [86, 125] and facilitates perception [18].

8.4 Requirements Validation

A correct definition of the requirements of a system has a great impact on the rest of the development process. Thus, to assist software engineers to firm up and explore requirements, several tools were proposed using 3D graphics as a way to validate them as early as possible. For example, *PNVIS* [76, 182] provides a 3D visualization that helps to analyze the behavior of a system modeled as Petri nets. *ScenarioML* [75] models a collection of scenarios as social interactions between agents by assigning a character to each actor and entity in these scenarios. During their interaction, actors and agents move to face each other, and speak their actions

Figure 16. ReqViz3D. Image courtesy of CRL Publishing Ltd. © 2005

as a means of expressing their accomplishment. Finally, *ReqViz3D* [74] is a tool to help users in the process of requirements understanding and validation using 3D visualization techniques. This tool allows specifying the requirements in the formal language Z, defining a graphical representation of them and creating a 3D animated visualization of their execution through which the users can validate them (see Figure 16). To sum up, the application of the third dimension in this area helps to provide familiarity, realism and real world representations [74, 75].

8.5 Other Purposes

Software visualization has also been applied for other purposes such as ontology visualization and semantic web [80, 183], and cyber attacks [79]. For example, Bosca et al. [80, 183] proposed a solution (*Ontosphere3D*) for the visualization and exploration of ontologies using a 3-dimensional space, as shown in Figure 17. They also reused the visualization component for supporting semantic web applications. Lastly, *Starmine* [79] is a cyber attack monitoring system that helps system administrators to analyze threats and make a right decision. Using 3D graphics, the system integrates different typical views of cyber threats (see Figure 18): geographical view (to show statistical information of cyber attacks, and geographical location of the source and the destination of each attack), logical view (to see automatic attacks such as scans or Internet worms) and temporal view (to understand transitions of the attacks).
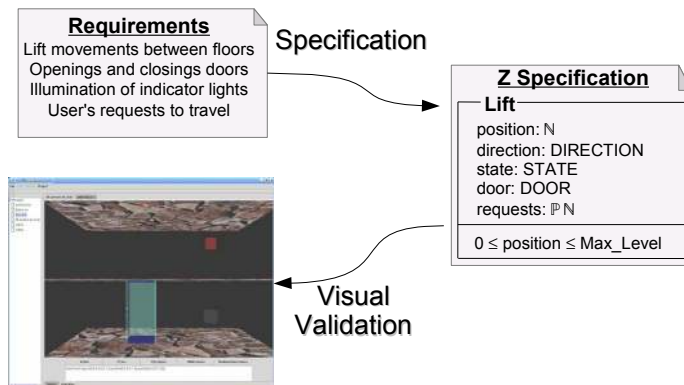
9 CONCLUSIONS

The trend towards more and more visual information is accelerating. There is an explosion of new software visualization techniques that improve software abstractions, create new representations based on real or abstract metaphors, make better layout algorithms, or develop

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

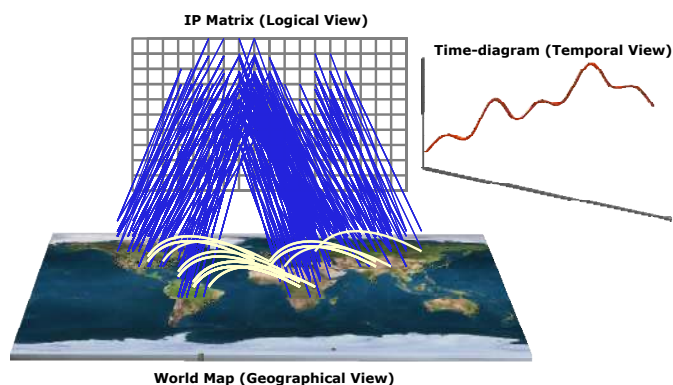IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS

15

Figure 18. Visualizing cyber attacks

new navigation mechanisms to browse through complex software representations. Indeed, current advances in visualization are providing new ways to look at software helping software engineers to comprehend, design and analyze software. However, substantial research is still needed to make software tools useful and not just research prototypes. Considering current approaches, discussions and challenges in information and software visualization, and our own experiences as well, we have identified the following areas of research and improvement:

- **Usability**: Software developers want to do their development as quickly and as better as possible. However, many approaches just focus on creating new visualization techniques with little attention on user needs and capabilities. In fact, human factors should play an important role in the design and evaluation of software visualization tools [187].

- **Computer supported collaborative visualization**: Software projects are inherently cooperative requiring many stakeholders to coordinate their activities to produce a software system. In particular, during the last years, software development has also become a geographically distributed activity [188]. Thus, research in collaborative visualization [189] with focus on software may help to improve software engineering process. For example, current technology provided by 3D collaborative virtual worlds for gaming and social interaction may support new ways of working, learning and visualizing software [100, 190].

- **Integration**: Many 3D software visualization systems have been already produced, most of them academic research projects. However, effective SV techniques should be integrated into real working environments, such as *Eclipse* [176, 180].

- **New display technologies**: As new devices such as high-resolution displays [191] or mobile devices [192] became more common, new techniques must be created or adapted to support device capabilities [134, 193, 194]. For example, take advantage

of large screens to simultaneously display more data, but taking into account that some of it will be outside the user's focal visual field.

- **Widely availability of 3D advanced displays and interactive facilities**: One of the main difficulties of 3D is navigation caused by the discrepancy of using 2D screens and 2D input devices to interact with a 3D world, combined with missing motion and stereo cues [38, 103, 106]. Once we turn into post-WIMP interfaces and adopt specialized hardware (e.g., stereo views, haptic displays, gaze, and motion tracking), 3D techniques may have a substantial effect on software visualization [7].

ACKNOWLEDGEMENTS

REFERENCES

[1] R. Mili and R. Steiner, "Software engineering - introduction," in *Revised Lectures on Software Visualization, International Seminar*. London, UK: Springer-Verlag, 2002, pp. 129–137.

[2] M. E. Tudoreanu, "Designing effective program visualization tools for reducing user's cognitive effort," in *SoftVis '03: Proceedings of the 2003 ACM Symposium on Software visualization*. New York, NY, USA: ACM Press, 2003, pp. 105–ff.

[3] S. Card, J. MacKinlay, and B. Shneiderman, Eds., *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann Publishers, 1998.

[4] C. Ware, *Information visualization: perception for design*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. - Elsevier, 2004.

[5] C. Chen, *Information Visualization: Beyond the Horizon*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[6] J. T. Stasko, M. H. Brown, and B. A. Price, Eds., *Software Visualization: Programming as a Multimedia Experience*. Cambridge, MA, USA: MIT Press, 1997.

[7] S. Diehl, *Software Visualization: Visualizing the Structure, Behaviour, and Evolution of Software*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007.

[8] M. L. Staples and J. M. Bieman, "3-D visualization of software structure," *Advances in Computer*, vol. 49, 1999.

[9] K. Zhang, Ed., *Software Visualization: From Theory to Practice*. Norwell, MA, USA: Kluwer Academic Publishers, 2003.

[10] D. Gracanin, K. Matkovic, and M. Eltoweissy, "Software visualization," *Innovations in Systems and Software Engineering, A NASA Journal*, vol. 1, no. 2, pp. 221–230, Sep. 2005.

[11] C. Knight, "System and software visualisation," in *Handbook of Software Engineering and Knowledge Engineering*. World Scientific Publishing Company, 2000.

[12] B. A. Myers, "Visual programming, programming by example, and program visualization: a taxonomy," in *CHI '86: Proceedings of the SIGCHI Conference on Human factors in Computing Systems*. New York, NY, USA: ACM Press, 1986, pp. 59–66.

[13] G.-C. Roman and K. C. Cox, "A taxonomy of program visualization systems," *Computer*, vol. 26, no. 12, pp. 11–24, 1993.

[14] B. Price, R. Baecker, and I. Small, "A principled taxonomy of software visualization," *Journal of Visual Languages and Computing*, vol. 4, no. 3, pp. 211–266, 1993.

[15] J. I. Maletic, A. Marcus, and M. Collard, "A task oriented view of software visualization," in *Proceedings of the 1st Workshop on Visualizing Software for Understanding and Analysis (VISSOFT 2002)*. IEEE, Jun. 2002, pp. 32–40.

[16] K. Gallagher, A. Hatch, and M. Munro, "A framework for software architecture visualization assessment," in *VISSOFT*. IEEE CS, Sep. 2005, pp. 76–81.

[17] I. Spence, "Visual psychophysics of simple graphical elements," *Journal of Experimental Psychology: Human Perception and Performance*, vol. 4, no. 16, pp. 683–692, 1990.

[18] P. Irani and C. Ware, "Diagramming information structures using 3d perceptual primitives," *ACM Trans. Comput.-Hum. Interact.*, vol. 10, no. 1, pp. 1–19, 2003.

[19] R. Brath, M. Peters, and R. Senior, "Visualization for communication: The importance of aesthetic sizzle," in *IV '05: Proceedings of the Ninth International Conference on Information Visualisation (IV'05)*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 724–729.

[20] J. T. Stasko and J. F. Wehrli, "Three-dimensional computation visualization," in *Proc. IEEE Symp. Visual Languages, VL*, E. P. Glinert and K. A. Olsen, Eds. IEEE Press, 24–27 Aug. 1993, pp. 100–107.

[21] A. Marcus, L. Feng, and J. I. Maletic, "3d representations for software visualization," in *SoftVis '03: Proceedings of the 2003 ACM symposium on Software visualization*. New York, NY, USA: ACM, 2003, pp. 27–ff.

[22] B. Shneiderman, "Why not make interfaces better than 3d reality?" *IEEE Comput. Graph. Appl.*, vol. 23, no. 6, pp. 12–15, 2003.

[23] P. Young and M. Munro, "Visualizing software in virtual reality," in *IWPC '98: Proceedings of the 6th International Workshop on Program Comprehension*. Washington, DC, USA: IEEE Computer Society, 1998, p. 19.

[24] E. R. Tufte, *Visual Explanations: Images and Quantities, Evidence and Narrative*. Cheshire, CT, USA: Graphics Press, 1997.

[25] M. B. McGrath and J. R. Brown, "Visual learning for science and engineering," *IEEE Comput. Graph. Appl.*, vol. 25, no. 5, pp. 56–63, 2005.

[26] D. M. Butler, J. C. Almond, R. D. Bergeron, K. W. Brodlie, and A. B. Haber, "Visualization reference models," in *Proceedings of the Visualization '93 Conference*, G. M. Nielson and D. Bergeron, Eds. San Jose, CA: IEEE Computer Society Press, Oct. 1993, pp. 337–342.

[27] I. Russell M. Taylor, "Practical scientific visualization examples," *SIGGRAPH Comput. Graph.*, vol. 34, no. 1, pp. 74–79, 2000.

[28] K. Einsfeld, A. Ebert, and J. Wolle, "Hannah: A vivid and flexible 3d information visualization framework," in *IV '07: Proceedings of the 11th International Conference Information Visualization*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 720–725.

[29] M. Tory and T. Moller, "Rethinking visualization: A high-level taxonomy," in *INFOVIS '04: Proceedings of the IEEE Symposium on Information Visualization (INFOVIS'04)*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 151–158.

[30] E. Carson, I. Parberry, and B. Jensen, "Algorithm explorer: visualizing algorithms in a 3d multimedia environment," in *SIGCSE '07: Proceedinds of the 38th SIGCSE technical Symposium on Computer science education*. New York, NY, USA: ACM Press, 2007, pp. 155–159.

[31] H. Gall, M. Jazayeri, and C. Riva, "Visualizing software release histories: The use of color and third dimension," in *ICSM '99: Proceedings of the IEEE International Conference on Software Maintenance*. Washington, DC, USA: IEEE Computer Society, 1999, p. 99.

[32] J. Rilling and S. P. Mudur, "On the use of metaballs to visually map source code structures and analysis results onto 3d space," in *WCRE '02*. Washington, DC, USA: IEEE Computer Society, 2002, p. 299.

[33] M. Tory, A. E. Kirkpatrick, and M. S. Atkins, "Visualization task performance with 2d, 3d, and combination displays," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 1, pp. 2–13, 2006, member-Torsten Moller.

[34] S. Baumgartner, A. Ebert, M. Deller, and S. Agne, "2d meets 3d: a human-centered interface for visual data exploration," in *CHI '07: CHI '07 extended abstracts on Human factors in computing systems*. New York, NY, USA: ACM Press, 2007, pp. 2273–2278.

[35] E. R. Tufte, *The Visual Display of Quantitative Information*. Cheshire, Connecticut, U.S.A.: Graphics Press, 1983.

[36] J. Wen, "Exploiting orthogonality in three dimensional graphics for visualizing abstract data," Brown University - Department of Computer Science, Technical Report CS-95-20, Jun. 1995.

[37] G. Robertson, S. K. Card, and J. D. Mackinlay, "Information visualization using 3D interactive animation," *Communications of the ACM*, vol. 36, no. 4, pp. 57–71, Apr. 1993.

[38] C. Ware and G. Franck, "Evaluating stereo and motion cues for visualizing information nets in three dimensions," *ACM Trans. Graph.*, vol. 15, no. 2, pp. 121–140, 1996.

[39] J. D. Mackinlay, G. G. Robertson, and S. K. Card, "The perspective wall: Detail and context smoothly integrated," in *Proceedings of ACM CHI'91 Conference on Human Factors in Computing Systems*, ser. Information Visualization, 1991, pp. 173–179.

[40] H. Koike, "The role of another spatial dimension in software visualization," *ACM Transactions on Information Systems*, vol. 11, no. 3, pp. 266–286, 1993.

[41] P. Strauss, "Iris inventor, a 3d graphics toolkit," in *OOPSLA '93*, 1993, pp. 192–200.

[42] H. Sowizral, K. Rushforth, and M. Deering, *The Java 3D API Specification*. Addison-Wesley, 1998.

[43] E. Parris and K. William, "Application graphics modeling support through object orientation," *IEEE Computer*, vol. 25, no. 10, pp. 84–90, 1992.

[44] S. P. Reiss, "An engine for the 3d visualization of program information," *Journal of Visual Languages and Computing*, vol. 6, no. 3, pp. 299–323, 1995.

[45] R. Dachselt, M. Hinz, and K. Meissner, "Contigra: an xml-based architecture for component-oriented 3d applications," in *Web3D '02*. New York, NY, USA: ACM Press, 2002, pp. 155–163.

[46] K. Larry and W. Wayne, "Groop: An object oriented toolkit for animated 3d graphics," in *OOPSLA'93*, 1993, pp. 309–325.

[47] Web3D-Consortium, "Communicating with real-time 3d across applications, networks, and xml web services," http://www.web3d.org, 2005.

[48] VRML-Consortium, "The virtual reality modeling language," http://www.web3d.org/technicalinfo/specifications/vrml97/index.htm, 1997.

[49] T. Panas, R. Lincke, and W. Lowe, "Online-configuration of software visualizations with vizz3d," in *SoftVis '05: Proceedings of the 2005 ACM Symposium on Software Visualization*. New York, NY, USA: ACM Press, 2005, pp. 173–182.

[50] J. F. Hopkins and P. A. Fishwick, "The rube framework for personalized 3-d software visualization," in *Revised Lectures on Software Visualization, International Seminar*. London, UK: Springer-Verlag, 2002, pp. 368–380.

[51] J. Heer, S. K. Card, and J. A. Landay, "prefuse: a toolkit for interactive information visualization," in *CHI '05: Proceedings of the SIGCHI Conference on Human factors in Computing Systems*. New York, NY, USA: ACM Press, 2005, pp. 421–430.

[52] A. Telea and L. Voinea, "A framework for interactive visualization of component-based software," in *EUROMICRO '04: Proceedings of the 30th EUROMICRO Conference (EUROMICRO'04)*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 567–574.

[53] K. P. Herndon, A. van Dam, and M. Gleicher, "The challenges of 3d interaction: a chi '94 workshop," *SIGCHI Bull.*, vol. 26, no. 4, pp. 36–43, 1994.

[54] D. A. Bowman, E. Kruijff, J. J. L. Jr., and I. Poupyrev, "An introduction to 3D user interface design," *Presence*, vol. 10, no. 1, pp. 96–108, 2001.

[55] C. Plaisant, J. Grosjean, and B. B. Bederson, "Spacetree: Supporting exploration in large node link tree, design evolution and empirical evaluation," *infovis*, vol. 00, p. 57, 2002.

[56] A. Cockburn and B. McKenzie, "Evaluating the effectiveness of spatial memory in 2d and 3d physical and virtual environments," in *CHI '02: Proceedings of the SIGCHI Conference on Human factors in Computing Systems*. New York, NY, USA: ACM, 2002, pp. 203–210.

[57] A. J. Hanson, E. A. Wernert, and S. B. Hughes, "Constrained navigation environments," *dagstuhl*, vol. 00, p. 95, 1997.

[58] D. A. Bowman and L. F. Hodges, "User interface constraints for immersive virtual environment applications," Graphics, Visualisation and Usability Center, Georgia Institute of Technology, USA, Tech. Rep. 95-26, 1995.

[59] S. Hughes and M. Lewis, "Robotic camera control for remote exploration," in *CHI '04: Proceedings of the SIGCHI Conference on*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS

17

Human Factors in Computing Systems. New York, NY, USA: ACM Press, 2004, pp. 511–517.

[60] A. Ahmed and P. Eades, "Automatic camera path generation for graph navigation in 3d," in *APVis '05: Proceedings of the 2005 Asia-Pacific Symposium on Information Visualisation*. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2005, pp. 27–32.

[61] R. P. Darken and J. L. Sibert, "A toolset for navigation in virtual environments," in *Proceedings of the ACM Symposium on User Interface Software and Technology*, 1993, pp. 157–165.

[62] R. Ingram and S. Benford, "Legibility enhancement for information visualisation," in *IEEE Visualization '95*, G. M. Nielson and D. Silver, Eds. IEEE Computer Soc. Press, 29–30 Oct. 1995, pp. 209–216.

[63] N. G. Vinson, "Design guidelines for landmarks to support navigation in virtual environments," in *CHI '99: Proceedings of the SIGCHI Conference on Human factors in Computing Systems*. New York, NY, USA: ACM Press, 1999, pp. 278–285.

[64] M. C. Chuah, S. F. Roth, J. Mattis, and J. Kolojejchick, "SDM: Selective dynamic manipulation of visualizations," in *Proceedings of the ACM Symposium on User Interface Software and Technology*, ser. 3D User Interfaces, 1995, pp. 61–70.

[65] K. Mullet, D. L. Schiano, G. Robertson, J. Tesler, B. Tversky, K. Mullet, and D. J. Schiano, "3d or not 3d: More is better or less is more?" in *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*, ser. Panels, vol. 2, 1995, pp. 174–175.

[66] R. Koschke, "Software visualization for reverse engineering," in *Software Visualization*, S. Diehl, Ed. Springer Verlag, 2002, vol. 2269 of LNCS State-of-the-Art Survey.

[67] S. Diehl, Ed., *Software Visualization*. Springer Verlag, 2002, vol. 2269 of LNCS State-of-the-Art Survey.

[68] G. Parker, G. Franck, and C. Ware, "Visualization of large nested graphs in 3D: Navigation and interaction," *J. Vis. Lang. Comput*, vol. 9, no. 3, pp. 299–317, 1998.

[69] Bentley and Kernighan, "System for algorithm animation," *Comput Syst*, vol. 4, no. 1, pp. 5–30, 1991.

[70] G.-C. Roman, K. C. Cox, C. D. ox, and J. Y. Plun, "Pavane: A system for declarative visualization of concurrent computations," *Journal of Visual Languages and Computing*, vol. 3, pp. 161–193, 1992.

[71] M. H. Brown and M. A. Najork, "Algorithm animation using 3d interactive graphics," in *UIST '93: Proceedings of the 6th Annual ACM Symposium on User interface software and technology*. New York, NY, USA: ACM Press, 1993, pp. 93–100.

[72] H. Koike, T. Takada, and T. Masui, "Visualinda: A framework for visualizing parallel linda programs," in *VL '97: Proceedings of the 1997 IEEE Symposium on Visual Languages (VL '97)*. Washington, DC, USA: IEEE Computer Society, 1997, p. 174.

[73] K. Osawa, N.and Asai, M. Suzuki, Y. Sugimoto, and F. Saito, "An immersive programming system: Ougi," in *Proceedings of the 12th International Conference on Artificial Reality and Telexistence (ICAT 2002)*, 2002, pp. 36–43.

[74] A. R. Teyseyre and M. Campo, "Early requirements validation with 3d worlds," *Computer Systems Science & Engineering*, vol. Automated Tools for Requirements Engineering, no. 20, pp. 61–72, January 2005.

[75] T. A. Alspaugh, B. Tomlinson, and E. Baumer, "Using social agents to visualize software scenarios," in *SoftVis '06: Proceedings of the 2006 ACM Symposium on Software visualization*. New York, NY, USA: ACM Press, 2006, pp. 87–94.

[76] H. Giese, E. Kindler, F. Klein, and R. Wagner, "Reconciling scenario-centered controller design with state-based system models," in *SCESM '05: Proceedings of the Fourth International Workshop on Scenarios and state machines: models, algorithms and tools*. New York, NY, USA: ACM Press, 2005, pp. 1–5.

[77] H. Koike and H.-C. Chu, "Vrcs: Integrating version control and module management using interactive 3d graphics," in *VL '97: Proceedings of the 1997 IEEE Symposium on Visual Languages (VL '97)*. Washington, DC, USA: IEEE Computer Society, 1997, p. 168.

[78] X. Xie, D. Poshyvanyk, and A. Marcus, "Visualization of cvs repository information," in *WCRE '06: Proceedings of the 13th Working Conference on Reverse Engineering (WCRE 2006)*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 231–242.

[79] Y. Hideshima and H. Koike, "Starmine: a visualization system for cyber attacks," in *APVis '06: Proceedings of the 2006 Asia-Pacific Symposium on Information Visualisation*. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2006, pp. 131–138.

[80] A. Bosca, D. Bonino, M. Comerio, S. Grega, and F. Corno, "A reusable 3d visualization component for the semantic web," in *Web3D '07: Proceedings of the Twelfth International Conference on 3D web technology*. New York, NY, USA: ACM Press, 2007, pp. 89–96.

[81] Y. Frishman and A. Tal, "Visualization of mobile object environments," in *SoftVis '05: Proceedings of the 2005 ACM Symposium on Software Visualization*. New York, NY, USA: ACM, 2005, pp. 145–154.

[82] J. I. Maletic, A. Marcus, and L. Feng, "Source viewer 3d (sv3d): a framework for software visualization," in *ICSE '03: Proceedings of the 25th International Conference on Software Engineering*. Washington, DC, USA: IEEE Computer Society, 2003, pp. 812–813.

[83] D. Bonyuet, M. Ma, and K. Jaffrey, "3d visualization for software development," in *ICWS '04: Proceedings of the IEEE International Conference on Web Services (ICWS'04)*. Washington, DC, USA: IEEE Computer Society, 2004, p. 708.

[84] P. McIntosh, M. Hamilton, and R. G. van Schyndel, "X3D-UML: enabling advanced UML visualisation through X3D," in *Proceedings of the Tenth International Conference on 3D Web Technology*, N. W. John, S. Ressler, L. Chittaro, and D. A. Duce, Eds. ACM, 2005, pp. 135–142.

[85] M. Balzer and O. Deussen, "Hierarchy based 3d visualization of large software structures," in *VIS '04: Proceedings of the Conference on Visualization '04*. Washington, DC, USA: IEEE Computer Society, 2004, p. 598.4.

[86] O. Greevy, M. Lanza, and C. Wysseier, "Visualizing live software systems in 3d," in *SoftVis '06: Proceedings of the 2006 ACM Symposium on Software Visualization*. New York, NY, USA: ACM Press, 2006, pp. 47–56.

[87] O. Radfelder and M. Gogolla, "On better understanding uml diagrams through interactive three-dimensional visualization and animation," in *AVI '00: Proceedings of the Working Conference on Advanced visual interfaces*. New York, NY, USA: ACM Press, 2000, pp. 292–295.

[88] G. Langelier, H. Sahraoui, and P. Poulin, "Visualization-based analysis of quality for large-scale software systems," in *ASE '05: Proceedings of the 20th IEEE/ACM International Conference on Automated software engineering*. New York, NY, USA: ACM, 2005, pp. 214–223.

[89] J.-Y. Vion-Dury and M. Santana, "Virtual images: interactive visualization of distributed object-oriented systems," in *OOPSLA '94: Proceedings of the Ninth Annual Conference on Object-oriented programming systems, language, and applications*. New York, NY, USA: ACM Press, 1994, pp. 65–84.

[90] L. Feijs and R. D. Jong, "3D visualization of software architectures," *Communications of the ACM*, vol. 41, no. 12, pp. 73–78, Dec. 1998.

[91] M. Denford, T. O'Neill, and J. Leaney, "Architecture-based visualisation of computer based systems," in *ECBS '02: Proceedings of the 9th IEEE International Conference on Engineering of Computer-Based Systems*. Washington, DC, USA: IEEE Computer Society, 2002, pp. 139–146.

[92] J. Rekimoto and M. Green, "The information cube: Using transparency in 3d information visualization," in *Proc. 3rd Annual Workshop on Information Technologies and Systems*, 1993, pp. 125–132.

[93] T. Panas, R. Berrigan, and J. Grundy, "A 3d metaphor for software production visualization," in *IV '03: Proceedings of the Seventh International Conference on Information Visualization*. Washington, DC, USA: IEEE Computer Society, 2003, p. 314.

[94] S. M. Charters, C. Knight, N. Thomas, and M. Munro, "Visualisation for informed decision making; from code to components," in *SEKE '02: Proceedings of the 14th International Conference on Software engineering and knowledge engineering*. New York, NY, USA: ACM Press, 2002, pp. 765–772.

[95] C. Knight and M. Munro, "Virtual but visible software," in *IV '00: Proceedings of the International Conference on Information Visualisation*. Washington, DC, USA: IEEE Computer Society, 2000, p. 198.

[96] D. Ploix, "Building program metaphors," in *Proceedings of PPIG Workshop*, 1996, pp. 125–129.

[97] H. Graham, H. Y. Yang, and R. Berrigan, "A solar system metaphor for 3d visualisation of object oriented software metrics," in *APVis '04: Proceedings of the 2004 Australasian Symposium*

on Information Visualisation. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2004, pp. 53–59.

[98] B. A. Malloy and J. F. Power, "Using a molecular metaphor to facilitate comprehension of 3d object diagrams," in *VLHCC '05: Proceedings of the 2005 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC'05)*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 233–240.

[99] K. Kahn, "Drawings on napkins, video-game animation, and other ways to program computers," *Commun. ACM*, vol. 39, no. 8, pp. 49–59, 1996.

[100] B. Kot, B. Wuensche, J. Grundy, and J. Hosking, "Information visualisation utilising 3d computer game engines case study: a source code comprehension tool," in *CHINZ '05: Proceedings of the 6th ACM SIGCHI New Zealand Chapter's International Conference on Computer-human interaction*. New York, NY, USA: ACM, 2005, pp. 53–60.

[101] J. Mackinlay, "Automating the design of graphical presentations of relational information," *ACM Transactions on Graphics*, vol. 5, no. 2, pp. 110–141, Apr. 1986.

[102] G. D. Battista, P. Eades, R. Tamassia, and I. G. Tollis, *Graph Drawing: Algorithms for the Visualization of Graphs*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1998.

[103] I. Herman, G. Melançon, and M. S. Marshall, "Graph visualization and navigation in information visualization: A survey," *IEEE Transactions on Visualization and Computer Graphics*, vol. 6, no. 1, pp. 24–43, 2000.

[104] H.-J. Schulz and H. Schumann, "Visualizing graphs - a generalized view," in *IV '06: Proceedings of the Conference on Information Visualization*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 166–173.

[105] C. Ware and P. Mitchell, "Visualizing graphs in three dimensions," *ACM Trans. Appl. Percept.*, vol. 5, no. 1, pp. 1–15, 2008.

[106] ——, "Reevaluating stereo and motion cues for visualizing graphs in three dimensions," in *APGV '05: Proceedings of the 2nd Symposium on Applied perception in graphics and visualization*. New York, NY, USA: ACM, 2005, pp. 51–58.

[107] B. Shneiderman and A. Aris, "Network visualization by semantic substrates," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 733–740, Sept.-Oct. 2006.

[108] M.-A. Storey, C. Best, and J. Michaud, "Shrimp views: An interactive environment for exploring java programs," *iwpc*, vol. 00, p. 0111, 2001.

[109] M.-A. Storey, N. F. Noy, M. Musen, C. Best, R. Fergerson, and N. Ernst, "Jambalaya: an interactive environment for exploring ontologies," in *IUI '02: Proceedings of the 7th International Conference on Intelligent user interfaces*. New York, NY, USA: ACM, 2002, pp. 239–239.

[110] J. I. Maletic, A. Marcus, G. Dunlap, and J. Leigh, "Visualizing object-oriented software in virtual reality," *iwpc*, vol. 00, p. 0026, 2001.

[111] S. P. Reiss, "Bee/hive: A software visualization back end," in *Proceedings of ICSE 2001 Workshop on Software Visualization*, 2001, pp. 44–48.

[112] T. Munzner, "H3: laying out large directed graphs in 3d hyperbolic space," in *INFOVIS '97: Proceedings of the 1997 IEEE Symposium on Information Visualization (InfoVis '97)*. Washington, DC, USA: IEEE Computer Society, 1997, p. 2.

[113] C. Lewerentz and A. Noack, "Crococosmos – 3d visualization of large object-oriented programs," in *Graph Drawing Software*. Springer-Verlag, 2003, pp. 279–297.

[114] T. Dwyer, "Three dimensional uml using force directed layout," in *APVis '01: Proceedings of the 2001 Asia-Pacific Symposium on Information visualisation*. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2001, pp. 77–85.

[115] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Massachusetts: Addison Wesley, 1994.

[116] M. Campo, T. Price, and A. Teyseyre, "Uma abordagem 3d para a visualização de padrões de projeto," in *Anais XI Simpósio Brasileiro de Engenharia de Software*, Fortaleza, Brasil, October 1997.

[117] G. G. Robertson, J. D. Mackinlay, and S. K. Card, "Cone trees: Animated 3D visualizations of hierarchical information," in *Proceedings of ACM CHI'91 Conference on Human Factors in Computing Systems*, ser. Information Visualization, 1991, pp. 189–194.

[118] R. Berghammer and A. Fronk, "Applying relational algebra in 3D graphical software design," in *RelMiCS*, ser. Lecture Notes

in Computer Science, R. Berghammer, B. Möller, and G. Struth, Eds., vol. 3051. Springer, 2003, pp. 62–74.

[119] W. Wang, H. Wang, G. Dai, and H. Wang, "Visualization of large hierarchical data by circle packing," in *CHI '06: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. New York, NY, USA: ACM Press, 2006, pp. 517–520.

[120] F. van Ham and J. J. van Wijk, "Beamtrees : Compact visualization of large hierarchies," *infovis*, vol. 00, p. 93, 2002.

[121] B. Johnson and B. Shneiderman, "Tree-maps: a space-filling approach to the visualization of hierarchical information structures," in *VIS '91: Proceedings of the 2nd Conference on Visualization '91*. Los Alamitos, CA, USA: IEEE Computer Society Press, 1991, pp. 284–291.

[122] S. P. Reiss, "An overview of bloom," in *PASTE '01: Proceedings of the 2001 ACM SIGPLAN-SIGSOFT Workshop on Program analysis for software tools and engineering*. New York, NY, USA: ACM Press, 2001, pp. 2–5.

[123] S. G. Eick, J. L. Steffen, and J. Eric E. Sumner, "Seesoft-a tool for visualizing line oriented software statistics," *IEEE Trans. Softw. Eng.*, vol. 18, no. 11, pp. 957–968, 1992.

[124] P. Young, "Visualising software in cyberspace," Ph.D. Thesis, University of Durham, Oct. 1999.

[125] T. Panas, T. Epperly, D. Quinlan, A. Saebjornsen, and R. Vuduc, "Communicating software architecture using a unified single-view visualization," in *ICECCS '07: Proceedings of the 12th IEEE International Conference on Engineering Complex Computer Systems (ICECCS 2007)*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 217–228.

[126] R. Wettel and M. Lanza, "Program comprehension through software habitability," in *ICPC '07: Proceedings of the 15th IEEE International Conference on Program Comprehension*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 231–240.

[127] J. Wise, J. Thomas, K. Pennock, D. Lantrip, M. Pottier, A. Schur, and V. Crow, *Visualizing the non-visual: spatial analysis and interaction with information for text documents*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999, pp. 442–450.

[128] A. Celentano, M. Nodari, and F. Pittarello, "Adaptive interaction in web3d virtual worlds," in *Web3D '04: Proceedings of the Ninth International Conference on 3D Web technology*. New York, NY, USA: ACM Press, 2004, pp. 41–50.

[129] U. Wiss and D. A. Carr, "An empirical study of task support in 3d information visualizations," *iv*, vol. 00, p. 392, 1999.

[130] R. Dachselt and M. Hinz, "Three-dimensional widgets revisited—towards future standardization," in *New directions in 3D user interfaces*, D. Bowman, B. Froehlich, Y. Kitamura, and W. Stuerzlinger, Eds. Springer-Verlag, 2005, pp. 89–92.

[131] R. Dachselt and A. Hübner, "Virtual environments: Three-dimensional menus: A survey and taxonomy," *Comput. Graph.*, vol. 31, no. 1, pp. 53–65, 2007.

[132] B. Reitinger, D. Kranzlmuller, and J. Volkert, "The most immersive approach for parallel and distributed program analysis," *iv*, vol. 00, p. 0517, 2001.

[133] M. Tory and T. Moller, "Evaluating visualizations: Do expert reviews work?" *IEEE Comput. Graph. Appl.*, vol. 25, no. 5, pp. 8–11, 2005.

[134] A. Kerren, A. Ebert, and J. Meyer, "Introduction to human-centered visualization environments," in *Human-Centered Visualization Environments*, ser. LNCS Tutorial, A. Kerren, A. Ebert, and J. Meyer, Eds. Springer, 2007, vol. 4417, ch. 1, pp. 1–9.

[135] R. Kosara, C. G. Healey, V. Interrante, D. H. Laidlaw, and C. Ware, "User studies: Why, how, and when?" *IEEE Computer Graphics and Applications*, vol. 23, no. 4, pp. 20–25, 2003.

[136] P. Rheingans, "Are we there yet? exploring with dynamic visualization," *IEEE Computer Graphics and Applications*, vol. 22, no. 1, pp. 6–10, 2002.

[137] M. Winckler, P. Palanque, and C. Freitas, "Tasks and scenario-based evaluation of information visualization techniques," in *TAMODIA '04: Proceedings of the 3rd Annual Conference on Task models and diagrams*. New York, NY, USA: ACM Press, 2004, pp. 165–172.

[138] D. House, V. Interrante, D. Laidlaw, R. Taylor, and C. Ware, "Design and evaluation in visualization research," *vis*, vol. 00, p. 117, 2005.

[139] C. Plaisant, "The challenge of information visualization evaluation," in *AVI '04: Proceedings of the Working Conference on Advanced visual interfaces*. New York, NY, USA: ACM Press, 2004, pp. 109–116.

[140] G. Ellis and A. Dix, "An explorative analysis of user evaluation studies in information visualisation," in *BELIV '06: Proceedings of the 2006 AVI Workshop on BEyond time and errors*. New York, NY, USA: ACM Press, 2006, pp. 1–7.

[141] C. Chen and Y. Yu, "Empirical studies of information visualization: a meta-analysis," *Int. J. Hum.-Comput. Stud.*, vol. 53, no. 5, pp. 851–866, 2000.

[142] O. Kulyk, R. Kosara, J. Urquiza, and I. Wassink, "Human-centered aspects," in *Human-Centered Visualization Environments*, ser. LNCS Tutorial, A. Kerren, A. Ebert, and J. Meyer, Eds. Springer, 2007, vol. 4417, ch. 2, pp. 10–75.

[143] J. Stasko, A. Badre, and C. Lewis, "Do algorithm animations assist learning?: an empirical study and analysis," in *CHI '93: Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*. New York, NY, USA: ACM Press, 1993, pp. 61–66.

[144] D. Sun and K. Wong, "On evaluating the layout of uml class diagrams for program comprehension," *iwpc*, vol. 00, pp. 317–326, 2005.

[145] A. F. Blackwell, C. Britton, A. L. Cox, T. R. G. Green, C. A. Gurr, G. F. Kadoda, M. Kutar, M. Loomes, C. L. Nehaniv, M. Petre, C. Roast, C. Roe, A. Wong, and R. M. Young, "Cognitive dimensions of notations: Design tools for cognitive technology," in *CT '01: Proceedings of the 4th International Conference on Cognitive Technology*. London, UK: Springer-Verlag, 2001, pp. 325–341.

[146] M.-A. D. Storey, D. Cubranie, and D. M. German, "On the use of visualization to support awareness of human activities in software development: a survey and a framework," in *SoftVis '05: Proceedings of the 2005 ACM Symposium on Software Visualization*. New York, NY, USA: ACM Press, 2005, pp. 193–202.

[147] H. C. Purchase, L. Colpoys, M. McGill, and D. Carrington, "Uml collaboration diagram syntax: An empirical study of comprehension," *vissoft*, vol. 00, p. 13, 2002.

[148] C. F. J. Lange and M. R. V. Chaudron, "Interactive views to improve the comprehension of uml models - an experimental validation," in *ICPC '07: Proceedings of the 15th IEEE International Conference on Program Comprehension*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 221–230.

[149] S. Yusuf, H. Kagdi, and J. I. Maletic, "Assessing the comprehension of uml class diagrams via eye tracking," in *ICPC '07: Proceedings of the 15th IEEE International Conference on Program Comprehension*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 113–122.

[150] Y.-G. Guéhéneuc, "Taupe: towards understanding program comprehension," in *CASCON '06: Proceedings of the 2006 Conference of the Center for Advanced Studies on Collaborative Research*. New York, NY, USA: ACM, 2006, p. 1.

[151] P. Irani and C. Ware, "The effect of a perceptual syntax on the learnability of novel concepts," in *IV '04: Proceedings of the Information Visualisation, Eighth International Conference on (IV'04)*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 308–314.

[152] A. Marcus, D. Comorski, and A. Sergeyev, "Supporting the evolution of a software visualization tool through usability studies," in *IWPC '05: Proceedings of the 13th International Workshop on Program Comprehension*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 307–316.

[153] M.-A. D. Storey, K. Wong, H. A. Mueller, P. Fong, D. Hooper, and K. Hopkins, "On designing an experiment to evaluate a reverse engineering tool," in *WCRE '96: Proceedings of the 3rd Working Conference on Reverse Engineering (WCRE '96)*. Washington, DC, USA: IEEE Computer Society, 1996, p. 31.

[154] J. A. Jones and M. J. Harrold, "Empirical evaluation of the tarantula automatic fault-localization technique," in *ASE '05: Proceedings of the 20th IEEE/ACM International Conference on Automated software engineering*. New York, NY, USA: ACM Press, 2005, pp. 273–282.

[155] B. de Alwis, G. C. Murphy, and M. P. Robillard, "A comparative study of three program exploration tools," in *ICPC '07: Proceedings of the 15th IEEE International Conference on Program Comprehension*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 103–112.

[156] S. Bassil and R. K. Keller, "Software visualization tools: Survey and analysis," in *Proceedings IWPC 2001*, 2001, pp. 7–17.

[157] R. A. Amar and J. T. Stasko, "Knowledge precepts for design and evaluation of information visualizations," *IEEE Transactions on Visualization and Computer Graphics*, vol. 11, no. 4, pp. 432–442, 2005.

[158] C. North, "Toward measuring visualization insight," *IEEE Comput. Graph. Appl.*, vol. 26, no. 3, pp. 6–9, 2006.

[159] D. Shreiner, M. Woo, J. Neider, and T. Davis, *OpenGL(R) Programming Guide: The Official Guide to Learning OpenGL(R), Version 2 (5th Edition) (OpenGL)*. Addison-Wesley Professional, 2005.

[160] L. P. Deutsch, "Design Reuse and Frameworks in the Smalltalk-80 System," in *Software Reusability*, T. J. Biggerstaff and C. Richter, Eds. acm press, 1989, vol. II — Applications and Experience, ch. 3, pp. 57–71.

[161] J. Kelso, L. E. Arsenault, R. D. Kriz, and S. G. Satterfield, "Diverse: A framework for building extensible and reconfigurable device independent virtual environments," in *VR '02: Proceedings of the IEEE Virtual Reality Conference 2002*. Washington, DC, USA: IEEE Computer Society, 2002, p. 183.

[162] A. Bierbaum, C. Just, P. Hartling, K. Meinert, A. Baker, and C. Cruz-Neira, "Vr juggler: A virtual platform for virtual reality application development," in *VR '01: Proceedings of the Virtual Reality 2001 Conference (VR'01)*. Washington, DC, USA: IEEE Computer Society, 2001, p. 89.

[163] X. Zhang and D. Gračanin, "From coarse-grained components to dve applications: a service- and component-based framework," in *Web3D '07*. New York, NY, USA: ACM Press, 2007, pp. 113–121.

[164] J.-D. Fekete, "The infovis toolkit," in *INFOVIS '04: Proceedings of the IEEE Symposium on Information Visualization (INFOVIS'04)*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 167–174.

[165] R. Orosco and M. Campo, "Mamp: A design model for object-oriented visualization systems," in *Engineering for Human Computer Interaction*, C. S. and D. P., Eds. Kluwer Academic, 1998.

[166] M. Campo, R. Orosco, and A. Teyseyre, "Automatic abstraction management in information visualization systems," in *IV '97: Proceedings of the IEEE Conference on Information Visualisation*. Washington, DC, USA: IEEE Computer Society, 1997, p. 50.

[167] M. Meyer, T. Gîrba, and M. Lungu, "Mondrian: an agile information visualization framework," in *SoftVis '06: Proceedings of the 2006 ACM Symposium on Software visualization*. New York, NY, USA: ACM Press, 2006, pp. 135–144.

[168] Q. Wang, W. Wang, R. Brown, K. Driesen, B. Dufour, L. Hendren, and C. Verbrugge, "Evolve: an open extensible software visualization framework," in *SoftVis '03: Proceedings of the 2003 ACM Symposium on Software Visualization*. New York, NY, USA: ACM Press, 2003, pp. 37–ff.

[169] A. Fronk and J. Schröder, "Effects - a framework for 3d software visualization," Poster IEEE International Conference on Automated Software Engineering (ASE 04), Sep. 2004.

[170] M. Campo and T. Price, "Luthier- building framework-visualization tools," in *Implementing Object-Oriented Application Frameworks: Frameworks at Work*, M. Fayad and R. Johnson, Eds. Wiley, 1999.

[171] W. Löwe and T. Panas, "Rapid construction of software comprehension tools," *International Journal of Software Engineering and Knowledge Engineering*, vol. 15, no. 6, pp. 905–1023, December 2005.

[172] R. I. Bull, M.-A. Storey, J.-M. Favre, and M. Litoiu, "An architecture to support model driven software visualization," in *ICPC '06: Proceedings of the 14th IEEE International Conference on Program Comprehension (ICPC'06)*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 100–106.

[173] M. Itoh and Y. Tanaka, "3d component-based visualization framework for generating simple 3d applications using web services," in *WI '06: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 823–830.

[174] Y. Okada and Y. Tanaka, "Intelligentbox: a constructive visual software development system for interactive 3d graphic applications," in *CA '95: Proceedings of the Computer Animation*. Washington, DC, USA: IEEE Computer Society, 1995, p. 114.

[175] R. I. Bull, C. Best, and M.-A. Storey, "Advanced widgets for eclipse," in *eclipse '04: Proceedings of the 2004 OOPSLA Workshop on Eclipse technology eXchange*. New York, NY, USA: ACM Press, 2004, pp. 6–11.

[176] A. Fronk, A. Bruckhoff, and M. Kern, "3d visualisation of code structures in java software systems," in *SoftVis '06: Proceedings of*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS

20

the 2006 ACM Symposium on Software visualization. New York, NY, USA: ACM Press, 2006, pp. 145–146.

[177] A. Tal and D. Dobkin, "Visualization of geometric algorithms," *IEEE Transactions on Visualization and Computer Graphics*, vol. 01, no. 2, pp. 194–204, 1995.

[178] M. Najork, "Web-based algorithm animation," in *DAC '01: Proceedings of the 38th Conference on Design Automation.* New York, NY, USA: ACM, 2001, pp. 506–511.

[179] H. Koike and H.-C. Chu, "How does 3d visualization work in software engineering?: empirical study of a 3d version/module visualization system," in *ICSE '98: Proceedings of the 20th International Conference on Software Engineering.* Washington, DC, USA: IEEE Computer Society, 1998, pp. 516–519.

[180] M. Burch, S. Diehl, and P. Weibgerber, "Visual data mining in software archives," in *SoftVis '05: Proceedings of the 2005 ACM Symposium on Software visualization.* New York, NY, USA: ACM, 2005, pp. 37–46.

[181] O. Greevy, M. Lanza, and C. Wysseier, "Visualizing feature interaction in 3-d," in *VISSOFT '05: Proceedings of the 3rd IEEE International Workshop on Visualizing Software for Understanding and Analysis.* Washington, DC, USA: IEEE Computer Society, 2005, p. 30.

[182] E. Kindler and C. Páles, "3D-visualization of petri net models: Concept and realization," in *Applications and Theory of Petri Nets 2004,* ser. Lecture Notes in Computer Science, J. Cortadella and W. Reisig, Eds., vol. 3099. Springer, 2004, pp. 464–473.

[183] A. Bosca and D. Bonino, "Ontosphere3d: A multidimensional visualization tool for ontologies," in *DEXA '06: Proceedings of the 17th International Conference on Database and Expert Systems Applications.* Washington, DC, USA: IEEE Computer Society, 2006, pp. 339–343.

[184] A. Kerren and J. T. Stasko, "Algorithm animation - introduction," in *Revised Lectures on Software Visualization, International Seminar.* London, UK: Springer-Verlag, 2002, pp. 1–15.

[185] M. H. Brown and M. Najork, "Algorithm animation using interactive 3d graphics," in *Software Visualization: Programming as a Multimedia Experience,* J. Stasko, J. Domingue, M. H. Brown, and B. A. Price, Eds. MIT Press, 1997, ch. 9, pp. 119–135.

[186] K. Dunsire, T. O'Neill, M. Denford, and J. Leaney, "The abacus architectural approach to computer-based system and enterprise evolution," in *ECBS '05: Proceedings of the 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'05).* Washington, DC, USA: IEEE Computer Society, 2005, pp. 62–69.

[187] M. Tory and T. Moller, "Human factors in visualization research," *IEEE Transactions on Visualization and Computer Graphics,* vol. 10, no. 1, pp. 72–84, 2004.

[188] B. Sengupta, S. Chandra, and V. Sinha, "A research agenda for distributed software development," in *ICSE '06: Proceedings of the 28th International Conference on Software Engineering.* New York, NY, USA: ACM, 2006, pp. 731–740.

[189] J. Wood, H. Wright, and K. Brodlie, "Collaborative visualization," in *VIS '97: Proceedings of the 8th Conference on Visualization '97.* Los Alamitos, CA, USA: IEEE Computer Society Press, 1997, pp. 253–ff.

[190] B. Damer, "Meeting in the ether," *interactions,* vol. 14, no. 5, pp. 16–18, 2007.

[191] G. Wallace, O. Anshus, P. Bi, H. Chen, Y. Chen, D. Clark, P. Cook, A. Finkelstein, T. Funkhouser, A. Gupta, M. Hibbs, K. Li, Z. Liu, R. Samanta, R. Sukthankar, and O. Troyanskaya, "Tools and applications for large-scale display walls," *IEEE Comput. Graph. Appl.,* vol. 25, no. 4, pp. 24–33, 2005.

[192] L. Chittaro, "Visualizing information on mobile devices," *Computer,* vol. 39, no. 3, pp. 40–45, 2006.

[193] P. Baudisch, "Interacting with large displays," *Computer,* vol. 39, no. 4, pp. 96–97, 2006.

[194] A. Bezerianos and R. Balakrishnan, "View and space management on large displays," *IEEE Computer Graphics and Applications,* vol. 25, no. 4, pp. 34–43, 2005.

Alfredo R. Teyseyre received a BSc degree in Systems Engineering and a Master degree in Systems Engineering from the UNICEN University, Tandil, Argentina, in 1997 and 2001, respectively. Currently he is a PhD candidate at the Department of Computer Science, UNICEN University, Tandil, Argentina. He is also an Adjunct Professor at Computer Science Department of the UNICEN University at Tandil, Argentina.

His research interests include software visualization, information visualization, software architecture and frameworks, and lightweight formal methods.

Marcelo R. Campo received a PhD. Degree in Computer Science in the Universidade Federal do Rio Grande do Sul, Porto Alegre, Brazil in 1997 and the Systems Engineer degree at the UNICEN University, Tandil, Argentina in 1988. Currently he is an Associate Professor at Computer Science Department and Head of the ISISTAN Research Institute of the UNICEN University at Tandil, Argentina. He is also a research fellow of the National Council for Scientific and Technical Research of Argentina (CONICET) and National Coordinator of the IT area. He has over 50 papers published in main conferences and journals about software engineering topics.

His research interests includes intelligent aided software engineering, software architecture and frameworks, agent technology and software visualization.