An Overview of 3D Software Visualization

Alfredo R. Teyseyre and Marcelo R. 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 that a software engineer has of a computer system. For 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 by 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, experience 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 runtime 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], [4], [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], [7], [8], [9], [10], [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], [13], [14], [15], [16]. However, as a result of hardware advances, many applications nowadays support 3D graphics capabilities. The inclusion of aesthetically appealing elements such as 3D graphics and animation not only increases the design's appeal, intuitiveness, and memorability of a visualization but also eases perception of the human visual system [17], [18], [19].

Although there is a debate on 2D versus 3D in the information visualization area [20], [21], [22], the use of 3D software visualization has the potential to aid in the development process. Three-dimensional 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 the 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 Fig. 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

- *A.R. Teyseyre is with the ISISTAN Research Institute, Facultad de Ciencias Exactas, Universidad Nacional del Centro de la Provincia de Buenos Aires, Campus Universitario, Paraje Arroyo Seco, (B7001BBO) Tandil, Bs. As., Argentina. E-mail: teyseyre@exa.unicen.edu.ar.*
- *M.R. Campo is with the ISISTAN Research Institute, Facultad de Ciencias Exactas, Universidad Nacional del Centro de la Provincia de Buenos Aires, Campus Universitario, Paraje Arroyo Seco, (B7001BBO) Tandil, Bs. As., Argentina, and with the National Council for Scientific and Technical Research of Argentina (CONICET), C1033AAJ Buenos Aires, Argentina. E-mail: mcampo@exa.unicen.edu.ar.*

Manuscript received 8 Oct. 2007; revised 4 Mar. 2008; accepted 11 June 2008; published online 24 June 2008.
Recommended for acceptance by J.T. Stasko.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-2007-10-0155.
Digital Object Identifier no. 10.1109/TVCG.2008.86.

1077-2626/09/$25.00 © 2009 IEEE Published by the IEEE Computer Society

Authorized licensed use limited to: UNIVERSITY OF WINDSOR. Downloaded on January 15, 2010 at 09:59 from IEEE Xplore. Restrictions apply.

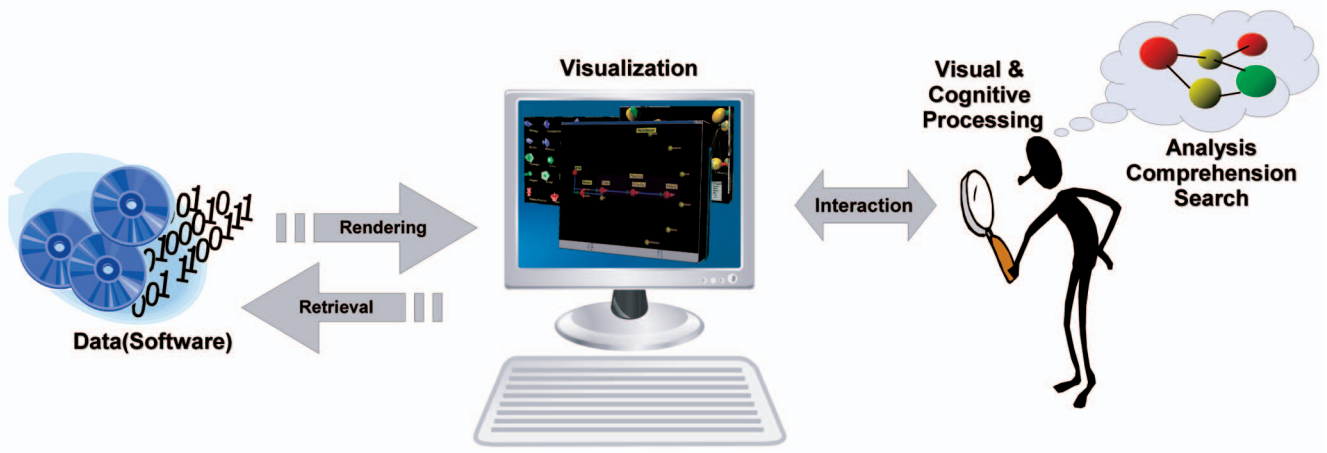


Fig. 1. Visualization process.

communication channel into human cognitive centers [4]. Furthermore, the visual system frees cognitive capacities by shifting part of the processing to it [3]. For example, as a result of visualization, scientists have changed their way of thinking, since they now say that they cannot do scientific research or communication without visualization [25]. Especially, visualization is a powerful tool that may help users to perform distinct types of cognitive processes [26]:

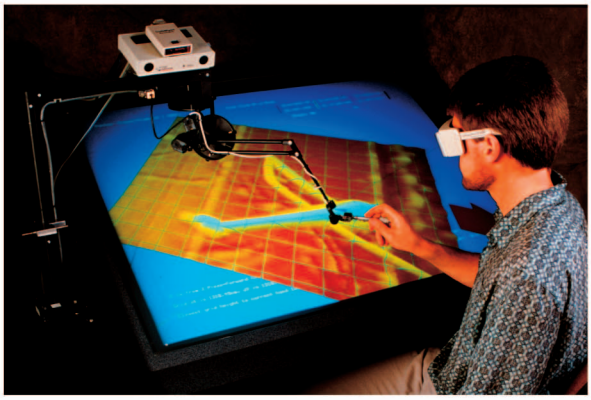
- ***Exploratory.*** The user does not know what he is looking for (Discovery).
- ***Analytical.*** The user knows what he is looking for in the data, trying to determine if it is there (Decision-making).
- ***Descriptive.*** The phenomenon represented in the data is known, but the user needs to have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

Visualization as a research field is categorized into two major subfields: ***Scientific Visualization*** and ***Information Visualization*** [3], [4]. Scientific visualization typically represents objects or concepts associated with phenomena from the physical world with an inherent spatial component (e.g., chemistry, meteorology, or the human body). For example, Fig. 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 nonspatial data, that is, abstract concepts and relationships (e.g., financial data, bibliographic sources, or software). For instance, Fig. 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].

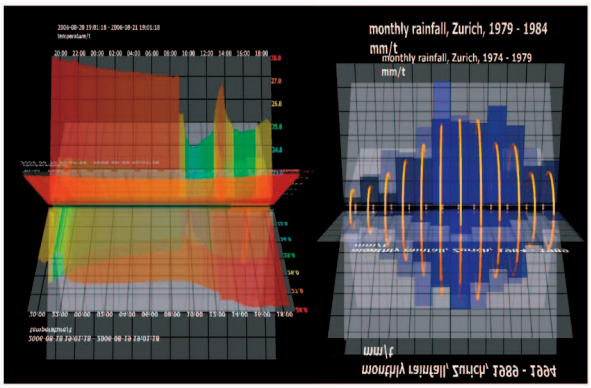
2.1 3D Visualization

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

As mentioned in Section 1, there is a controversial debate on 2D versus 3D in the information visualization area. In order to analyze and identify strengths and weaknesses of



(a)



(b)

Fig. 2. Scientific Visualization and Information Visualization examples. (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.

Authorized licensed use limited to: UNIVERSITY OF WINDSOR. Downloaded on January 15, 2010 at 09:59 from IEEE Xplore. Restrictions apply.

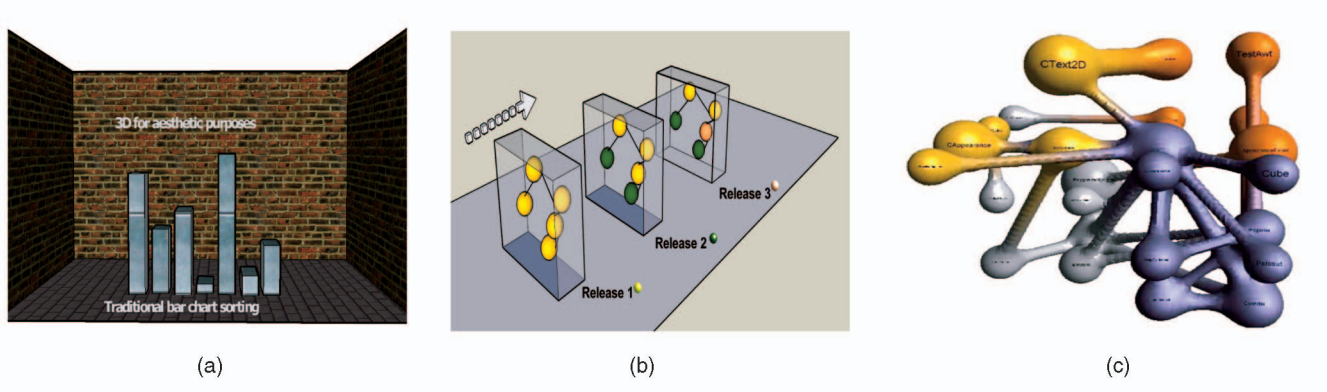


Fig. 3. Categorization of 3D view. (a) Augmented 2D view: a bubble sort visualization. (b) Adapted 2D view: software release history. (c) Inherent 3D view: Metaballs. Image courtesy of J. Rilling and S.P. Mudur. 2002 IEEE.

3D/2D, we first review a categorization of 3D visualizations that helps in the analysis [20]:

1. ***Augmented 2D views.*** This category includes typical 2D visualizations where the third dimension is added just for aesthetic purposes. For example, Fig. 3a shows a 3D presentation of a traditional 2D bar chart sorting algorithm [30].
2. ***Adapted 2D views.*** This category includes 2D visualizations extended to 3D to encode additional information. To illustrate this, Fig. 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 3D entities. For instance, Fig. 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 two 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 that is semantically richer. However, other authors think that 3D presentations facilitate the 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 the 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 2D 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 multiple 2D views in a single 3D view [18], [40]. Last, 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 2D 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], [42], [43], [44], [45], [46], 3D modeling languages [47], [48], or 3D software visualization frameworks [44], [49], [50], [51], [52].

However, one of the main problems of 3D applications is user adaptation. Most users just have experience with classical windows, icons, menus, pointing devices (*WIMP*) 2D 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 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

Authorized licensed use limited to: UNIVERSITY OF WINDSOR. Downloaded on January 15, 2010 at 09:59 from IEEE Xplore. Restrictions apply.

TABLE 1
3D Strengths and Weaknesses

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

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], [62], [63].

Finally, occlusion may distort the user's perception of the data space mainly when the information space is dense [64]. Especially, 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 versus 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 from 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, reengineering, and reverse engineering, 40 percent found software visualization very necessary for their work, and another 42 percent 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 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], [70], [71], debugging [72], 3D programming [73], requirements engineering [74], [75], [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], [85], [86], [87], [88], [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 not only 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 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 by using an abstract representation based on polycylinders [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 are based on *real-world* objects like 3D cities [93], [94], [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]. First, expressiveness criteria determine whether a visual representation can express the desired information. Second, 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].

Authorized licensed use limited to: UNIVERSITY OF WINDSOR. Downloaded on January 15, 2010 at 09:59 from IEEE Xplore. Restrictions apply.

Fig. 4. An example graph representing a software system.

These properties may be useful to create new visualizations or to evaluate existing ones.

4.1 Abstract Visual Representations

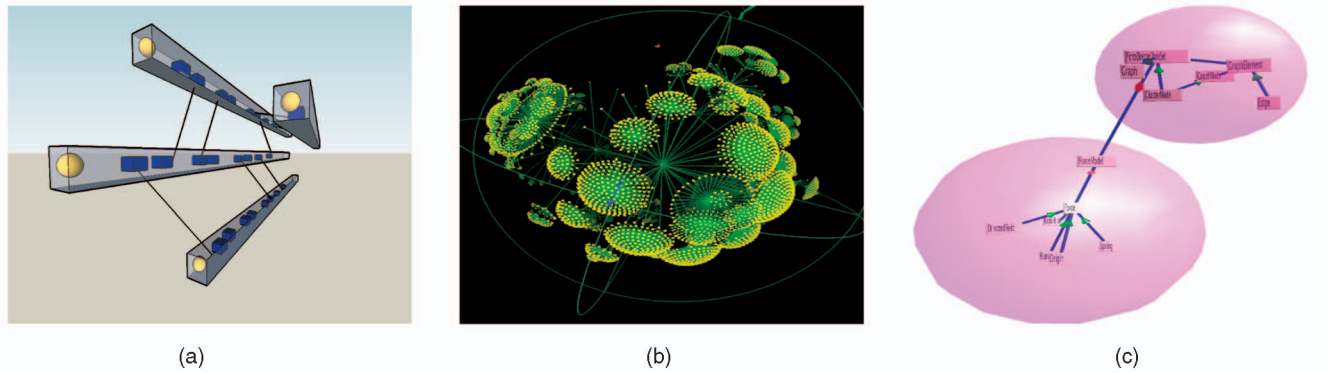
In this section, we describe several 3D abstract visual representations based on graphs, trees, and geometrical shapes. We also analyze their strengths and weaknesses.

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, Fig. 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]; for graph visualization in information visualization, see [103]; and for combined approaches, see [104].

Graph visualization in 2D space, representing software components around simple boxes and lines, has been applied. However, visualization may result to be completely incomprehensible, not only as software project complexity increases, but also when visualizing multiple 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 reevaluation 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 the source to the destination [107], [108], [109]. For instance, Shneiderman and Aris proposed a 2D approach using a layout strategy based on user-defined semantic substrates (nonoverlapping regions in which node placement is based on attributes) and interaction capabilities to control link visibility [107]. Despite 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, the layout of 3D graphs may actually differ. For example, aesthetic constraints such the 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 layout 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 instance, Fig. 5a shows a 2D layout of modules in a message passing system, where the third dimension displays the message sequencing



(a) (b) (c)

Fig. 5. Graph layout algorithms. (a) 3D dimension for time: message sequencing information. (b) Hyperbolic layout of the CVS repository produced by the 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.

Authorized licensed use limited to: UNIVERSITY OF WINDSOR. Downloaded on January 15, 2010 at 09:59 from IEEE Xplore. Restrictions apply.

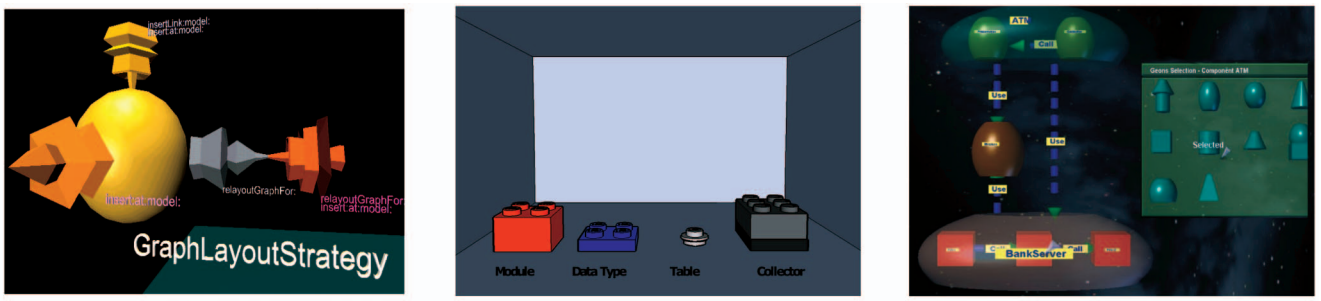


Fig. 6. Node representations. (a) Class and design patterns. (b) Lego bricks. (c) Geons.

information. Other techniques compose simultaneously several 2D visualizations using orthogonal axes [40]. In addition, 3D hyperbolic layouts [112] have been applied. This kind of layouts provides a distorted view of the graph, which makes it possible to interact with potentially large graphs, as shown in Fig. 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, Fig. 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 kinds 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, Fig. 6a presents a class visualized as volume composed of three semiaxes, 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* (Fig. 6b) [90] or 3D primitives called geons (Fig. 6c), which take advantage of the 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 noticeably, 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 the relationships between them such as inheritance or composition. Moreover, since trees have no cycles, unlike graphs, they are generally easy to lay out and interpret [3]. In general, tree visualization techniques encode hierarchical information using *node-link* diagrams, that is, explicit relations, and also *containment* representations, i.e., relations using space-filling methods. For instance, Fig. 7a shows a node-link representation [117], Fig. 7c shows a containment representation [92], and Fig. 7b shows 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 its 2D counterpart. The aim of this technique is to

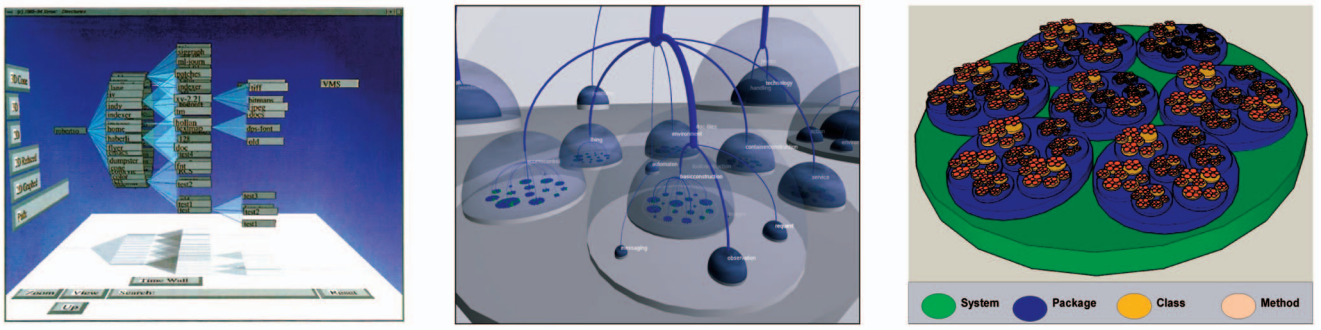


Fig. 7. Different tree representations. (a) Cone Tree. Image courtesy of G. G. Robertson, J. D. Mackinlay, and S. K. Card. Xerox PARC, Inc. (b) Hierarchical Net 3D. Image courtesy of M. Balzer and O. Deussen. 2004 IEEE. (c) Using Circle Data Packing to represent the software structure.

Authorized licensed use limited to: UNIVERSITY OF WINDSOR. Downloaded on January 15, 2010 at 09:59 from IEEE Xplore. Restrictions apply.

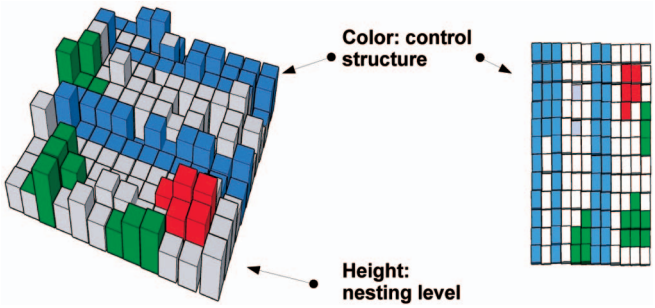


Fig. 8. sv3D representation versus 2D SeeSoft.

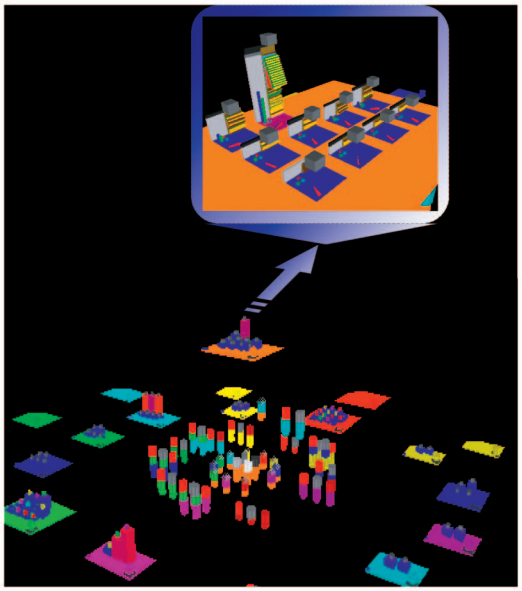
allow a greater amount of information to be navigated and displayed in an intuitive manner. Each subtree is laid out as a cone with its root at the top of the cone and the children along the cone base. In addition, nodes are semitransparent rectangles, and when one has been selected, the tree rotates, 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 creating a focal point on the structure and shifting 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 to be more effective than nested representations for the extraction of global hierarchical information (see Fig. 7c).

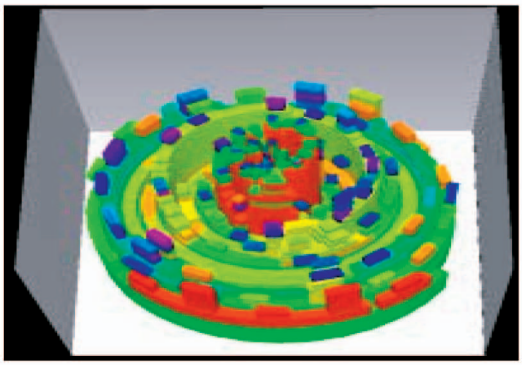
Finally, other approach called *Hierarchical Net 3D* [85] mixes containment and explicit relationships to show the static structure of object-oriented systems. The hierarchy of packages, classes, methods, and attributes is represented using nested hemispheres for packages, circles for classes, and boxes 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 3D geometrical shapes [21], [23], [89], [122].



(a)



(b)

Fig. 9. Abstract software representations. (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.

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 reengineering. 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, Fig. 8a shows a container that represents a source file, where each polycylinder represents a line of text from the source code associated with the container. In addition, color is used to represent the control structure, and height is used to represent the nesting level. On the other hand, a 2D *SeeSoft* representation (Fig. 8b) of the same file can just represent the 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 Fig. 9a). This system shows not only structural information but also runtime information using a technique called *Callstax* to represent the calling structure of

Authorized licensed use limited to: UNIVERSITY OF WINDSOR. Downloaded on January 15, 2010 at 09:59 from IEEE Xplore. Restrictions apply.

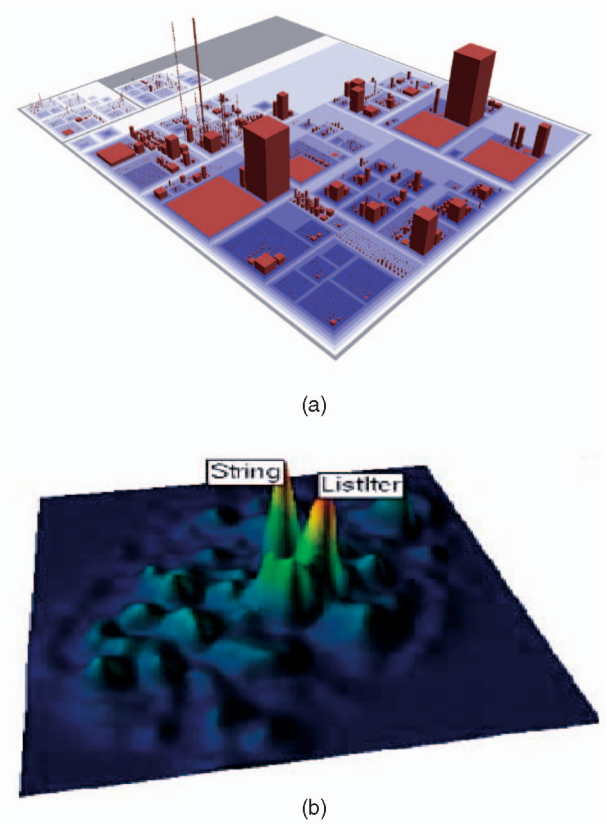


Fig. 10. Real-world metaphors. (a) ArgoUML as a 3D city. Image courtesy of R. Wettel and M. Lanza. 2007 IEEE. (b) 3D Landscape. Image courtesy of A. Telea and L. Voinea. 2004 IEEE.

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, Fig. 9b shows a sample trace, where color indicates the routine being called, height indicates the stack depth, and width indicates the total runtime.

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], [94], [95], [125], [126]. In particular, *CodeCity* [126] represents classes as buildings located in city districts, which, in turn, represent packages (see Fig. 10a). Another visualization [52] that helps to gain an overview of a software system is presented in Fig. 10b. It is based on a 3D landscape technique called *ThemeScapes* [127]. This visualization shows the relation "is called by" as a 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.

5 Interaction

Software visualization tools have to provide not only effective visual representations but also effective interaction styles to ease the exploration and help software engineers to achieve insight. However, interacting with 3D worlds is more complex than with 2D *WIMP* interfaces [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 the 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 (VEs), which is, in turn, intuitive as it 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]. Especially, 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, and 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, a simple six-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

Authorized licensed use limited to: UNIVERSITY OF WINDSOR. Downloaded on January 15, 2010 at 09:59 from IEEE Xplore. Restrictions apply.

zoomed in or out, but the position of the camera is fixed. In addition, several systems provide *semantic zoom*: the level of detail changes as the user inspects nodes closer. To be specific, as the viewpoint moves toward a particular object, at a certain threshold, the lowly detailed object representation will be replaced with a highly 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], [134], [135]. In addition, any usability evaluation of a visualization technique has to address both evaluations of visual representations and interaction styles, because exploration may help to achieve insight that a set of fixed images cannot [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], [134], [135], [138], [139], [140], [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 detailed 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 theoretical analytic studies. For example, Sun and Wong [144] evaluated two commercial UML tools using 14 criteria based on gestalt-theoretical 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], [13], [14], [15], [16], [146], which, in turn, provide support for the qualitative 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 Guehéneuc [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 other techniques. In addition, de 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], especially 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 programming 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

Authorized licensed use limited to: UNIVERSITY OF WINDSOR. Downloaded on January 15, 2010 at 09:59 from IEEE Xplore. Restrictions apply.

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 the 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 VEs or distributed VEs (DVEs) 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 3D applications, which are either stand alone or Web based.

Especially, 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. Another 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 prepackaged components such as data editing, filtering, rendering, and user actions. Finally, *vizz3D* [49], [171] is a framework for creating new software visualizations that enables us 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 everyday [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), and *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, the *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, Fig. 11 shows the animation of a geometric algorithm.

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

Authorized licensed use limited to: UNIVERSITY OF WINDSOR. Downloaded on January 15, 2010 at 09:59 from IEEE Xplore. Restrictions apply.

TABLE 2
Visualization Tool Summary

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

Authorized licensed use limited to: UNIVERSITY OF WINDSOR. Downloaded on January 15, 2010 at 09:59 from IEEE Xplore. Restrictions apply.

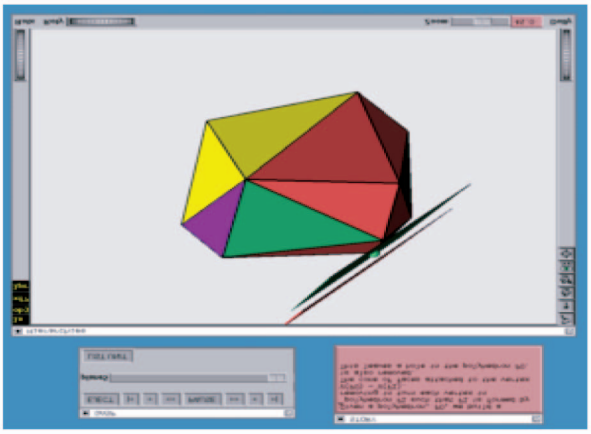


Fig. 11. GASP. Image courtesy of A. Tal and D. Dobkin. 1995 IEEE.

a certain release of the system is represented as a sphere, and versions composing the release are connected by links (see Fig. 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]. Last, another 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 levels of abstractions (i.e., source code [21], [82], detailed design [40], [52], [84], [85], [86], [87], [88], and software architectures [23], [90], [91]) and on different software views (i.e., software structure and/or its runtime behavior). For example, *sv3D* [21], [82] supports the visualization of large-scale software to assist in comprehension and analysis tasks at the code level. This tool represents a line of text from the source code using polycylinders. Different attributes such as color, position, height, and depth

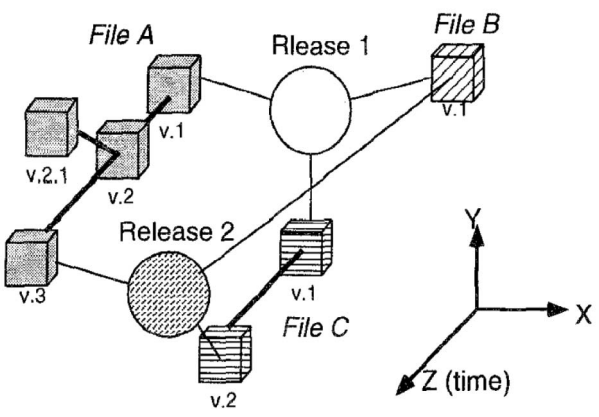


Fig. 12. 3D VRCS representation. Image courtesy of H. Koike and H. Chu. 1997 IEEE.

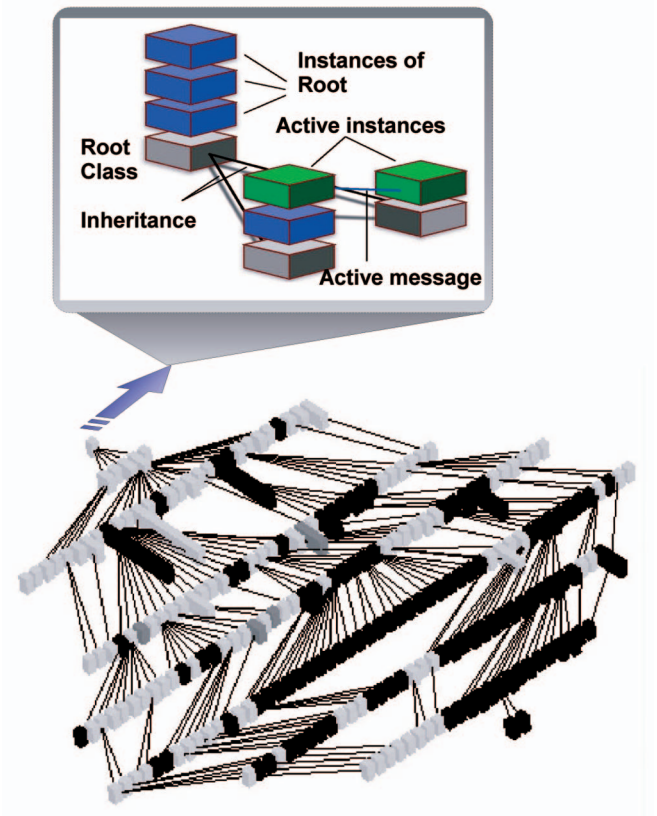


Fig. 13. TraceCrawler. Adapted from images courtesy of O. Greevy, M. Lanza, and C. Wysseier. 2005 IEEE.

can be associated with software properties and line-oriented metrics as well (e.g., color may reflect the control structure type, and height may reflect the 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 are mapped to lines. In particular, *TraceCrawler* [86], [181] helps to understand both the structure (i.e., classes and inheritance relationships) and the 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 Fig. 13). The user can map several metrics to the 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 us to see and communicate current development, quality, and costs of a software system quickly (Fig. 14). Also, another approach investigates how to improve the interpretation of

Authorized licensed use limited to: UNIVERSITY OF WINDSOR. Downloaded on January 15, 2010 at 09:59 from IEEE Xplore. Restrictions apply.

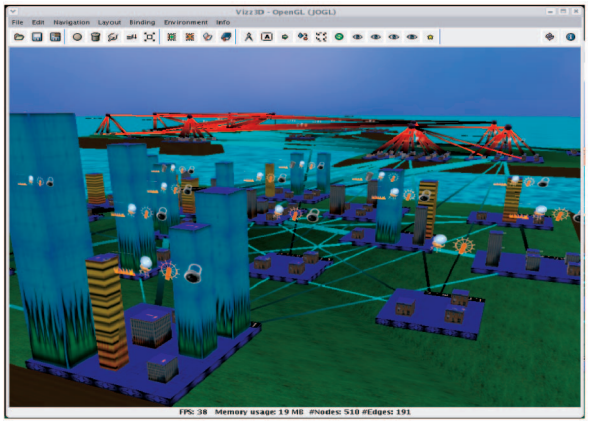


Fig. 14. Unified City. Image courtesy of T. Panas, T. Epperly, D. Quinlan, A. Saebjornsen, and R. Vuduc. 2007 IEEE.

node-link diagrams using 3D shaded elements instead of simple lines and outlines [18].

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 Fig. 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 metric-based information (e.g., Lines of Code (LOCs), 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

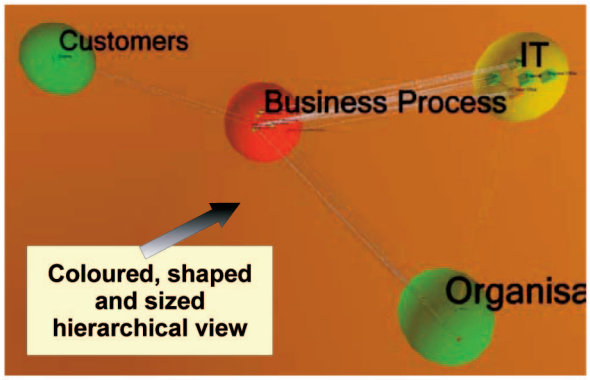


Fig. 15. ABACUS 3D Overall Enterprise Architecture View. The technique is "international patent pending" and has been implemented through the ABACUS product available from Avolution (www.avolution.com.au). Image courtesy of K. Dunsire, T. O'Neill, M. Denford, and J. Leaney. 2005 IEEE.

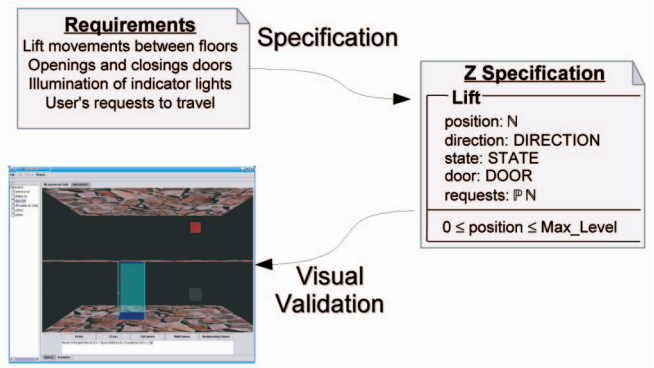


Fig. 16. ReqViz3D. Image courtesy of CRL Publishing Ltd. 2005.

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 as a means of expressing their accomplishment. Finally, *ReqViz3D* [74] is a tool to help users in the process of requirement 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 Fig. 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 3D space, as shown in Fig. 17. They also reused the visualization component for supporting semantic Web applications. Last, *Starmine* [79] is a cyber attack monitoring system that helps system administrators to analyze threats and make

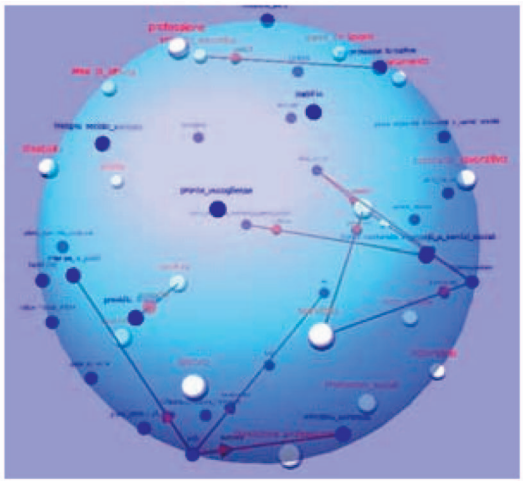


Fig. 17. OntoSphere3D: a multidimensional visualization tool for ontologies. Image courtesy of A. Bosca and D. Bonino. 2006 IEEE.

Authorized licensed use limited to: UNIVERSITY OF WINDSOR. Downloaded on January 15, 2010 at 09:59 from IEEE Xplore. Restrictions apply.

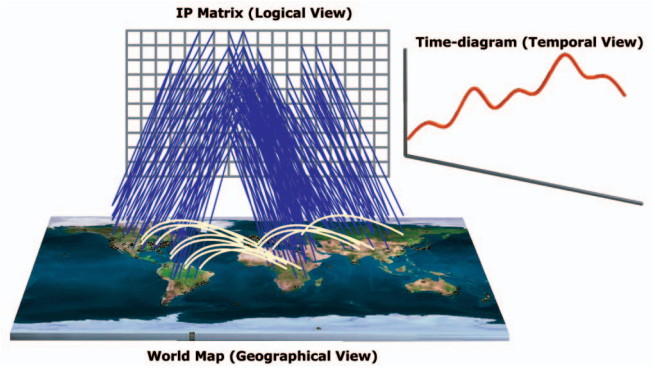


Fig. 18. Visualizing cyber attacks.

the right decision. Using 3D graphics, the system integrates different typical views of cyber threats (see Fig. 18): geographical view (to show the statistical information of cyber attacks and the 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 toward 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 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 for research and improvement:

- ***Usability.*** Software developers want to do their development as quickly and as good 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 the software engineering process. For example, the 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 being 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.
- ***Wide 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 them into post-WIMP interfaces and adopt specialized hardware (e.g., stereo views, haptic displays, gaze tracking, and motion tracking), 3D techniques may have a substantial effect on software visualization [7].

Acknowledgments

The authors would like to thank all the people and institutions that have allowed them to use many of the figures present in this paper. The anonymous reviewers deserve special thanks for their effort in making very detailed reviews. Finally, the authors thank the members of the ISISTAN Research Institute (http://www.exa.unicen.edu.ar/isistan/), Universidad Nacional del Centro de la Provincia de Buenos Aires (UNICEN), Tandil, Argentina.

References

[1] R. Mili and R. Steiner, "Software Engineering—Introduction," *Revised Lectures on Software Visualization, Int'l Seminar*, pp. 129-137, 2002.
[2] M.E. Tudoreanu, "Designing Effective Program Visualization Tools for Reducing User's Cognitive Effort," *Proc. ACM Symp. Software Visualization (SoftVis '03)*, p. 105-ff, 2003.
[3] S. Card, J. MacKinlay, and B. Shneiderman, eds., *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann, 1998.
[4] C. Ware, *Information Visualization: Perception for Design*. Morgan Kaufmann-Elsevier, 2004.
[5] C. Chen, *Information Visualization: Beyond the Horizon*. Springer, 2006.
[6] J.T. Stasko, M.H. Brown, and B.A. Price, eds., *Software Visualization: Programming as a Multimedia Experience*. MIT Press, 1997.
[7] S. Diehl, *Software Visualization: Visualizing the Structure, Behaviour, and Evolution of Software*. Springer, 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*. Kluwer Academic Publishers, 2003.
[10] D. Gracanin, K. Matkovic, and M. Eltoweissy, "Software Visualization," *Innovations in Systems and Software Eng., A NASA J.*, vol. 1, no. 2, pp. 221-230, Sept. 2005.
[11] C. Knight, "System and Software Visualisation," *Handbook of Software Eng. and Knowledge Eng.* World Scientific, 2000.
[12] B.A. Myers, "Visual Programming, Programming by Example, and Program Visualization: A Taxonomy," *Proc. ACM SIGCHI Conf. Human Factors in Computing Systems (CHI '86)*, pp. 59-66, 1986.

Authorized licensed use limited to: UNIVERSITY OF WINDSOR. Downloaded on January 15, 2010 at 09:59 from IEEE Xplore. Restrictions apply.

[13] G.-C. Roman and K.C. Cox, "A Taxonomy of Program Visualization Systems," *Computer*, vol. 26, no. 12, pp. 11-24, Dec. 1993.

[14] B. Price, R. Baecker, and I. Small, "A Principled Taxonomy of Software Visualization," *J. 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," *Proc. First Workshop Visualizing Software for Understanding and Analysis (VISSOFT '02)*, pp. 32-40, June 2002.

[16] K. Gallagher, A. Hatch, and M. Munro, "A Framework for Software Architecture Visualization Assessment," *Proc. Third Workshop Visualizing Software for Understanding and Analysis (VISSOFT '05)*, pp. 76-81, Sept. 2005.

[17] I. Spence, "Visual Psychophysics of Simple Graphical Elements," *J. 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. Computer-Human Interaction*, vol. 10, no. 1, pp. 1-19, 2003.

[19] R. Brath, M. Peters, and R. Senior, "Visualization for Communication: The Importance of Aesthetic Sizzle," *Proc. Ninth Int'l Conf. Information Visualisation (IV '05)*, pp. 724-729, 2005.

[20] J.T. Stasko and J.F. Wehrli, "Three-Dimensional Computation Visualization," *Proc. IEEE Symp. Visual Languages (VL '93)*, E.P. Glinert and K.A. Olsen, eds., pp. 100-107, Aug. 1993.

[21] A. Marcus, L. Feng, and J.I. Maletic, "3D Representations for Software Visualization," *Proc. ACM Symp. Software Visualization (SoftVis '03)*, p. 27-ff, 2003.

[22] B. Shneiderman, "Why Not Make Interfaces Better Than 3D Reality?" *IEEE Computer Graphics and Applications*, vol. 23, no. 6, pp. 12-15, Nov./Dec. 2003.

[23] P. Young and M. Munro, "Visualizing Software in Virtual Reality," *Proc. Sixth Int'l Workshop Program Comprehension (IWPC '98)*, p. 19, 1998.

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

[25] M.B. McGrath and J.R. Brown, "Visual Learning for Science and Eng.," *IEEE Computer Graphics and Applications*, vol. 25, no. 5, pp. 56-63, Sept./Oct. 2005.

[26] D.M. Butler, J.C. Almond, R.D. Bergeron, K.W. Brodlie, and A.B. Haber, "Visualization Reference Models," *Proc. Fourth IEEE Conf. Visualization*, G.M. Nielson and D. Bergeron, eds., pp. 337-342, Oct. 1993.

[27] I. Russell and M. Taylor, "Practical Scientific Visualization Examples," *ACM SIGGRAPH Computer Graphics*, 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," *Proc. 11th Int'l Conf. Information Visualization (IV '07)*, pp. 720-725, 2007.

[29] M. Tory and T. Moller, "Rethinking Visualization: A High-Level Taxonomy," *Proc. IEEE Symp. Information Visualization (InfoVis '04)*, pp. 151-158, 2004.

[30] E. Carson, I. Parberry, and B. Jensen, "Algorithm Explorer: Visualizing Algorithms in a 3D Multimedia Environment," *Proc. 38th Technical Symp. Computer Science Education (SIGCSE '07)*, pp. 155-159, 2007.

[31] H. Gall, M. Jazayeri, and C. Riva, "Visualizing Software Release Histories: The Use of Color and Third Dimension," *Proc. IEEE Int'l Conf. Software Maintenance (ICSM '99)*, p. 99, 1999.

[32] J. Rilling and S.P. Mudur, "On the Use of Metaballs to Visually Map Source Code Structures and Analysis Results onto 3D Space," *Proc. Ninth Working Conf. Reverse Eng. (WCRE '02)*, p. 299, 2002.

[33] M. Tory, A.E. Kirkpatrick, and M.S. Atkins, "Visualization Task Performance with 2D, 3D, and Combination Displays," *IEEE Trans. Visualization and Computer Graphics*, vol. 12, no. 1, pp. 2-13, Jan./Feb. 2006.

[34] S. Baumgartner, A. Ebert, M. Deller, and S. Agne, "2D Meets 3D: A Human-Centered Interface for Visual Data Exploration," *Extended Abstracts on Human Factors in Computing Systems (CHI '07)*, pp. 2273-2278, 2007.

[35] E.R. Tufte, *The Visual Display of Quantitative Information.* Graphics Press, 1983.

[36] J. Wen, "Exploiting Orthogonality in Three Dimensional Graphics for Visualizing Abstract Data," Technical Report CS-95-20, Dept. of Computer Science, Brown Univ., June 1995.

[37] G. Robertson, S.K. Card, and J.D. Mackinlay, "Information Visualization Using 3D Interactive Animation," *Comm. 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. Graphics*, 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," *Proc. ACM Conf. Human Factors in Computing Systems (CHI '91)*, pp. 173-179, 1991.

[40] H. Koike, "The Role of Another Spatial Dimension in Software Visualization," *ACM Trans. Information Systems*, vol. 11, no. 3, pp. 266-286, 1993.

[41] P. Strauss, "Iris Inventor, a 3D Graphics Toolkit," *Proc. Eighth Ann. Conf. Object-Oriented Programming Systems, Languages, and Applications (OOPSLA '93)*, pp. 192-200, 1993.

[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," *Computer*, vol. 25, no. 10, pp. 84-90, Oct. 1992.

[44] S.P. Reiss, "An Engine for the 3D Visualization of Program Information," *J. 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," *Proc. Seventh Int'l Conf. 3D Web Technology (Web3D '02)*, pp. 155-163, 2002.

[46] K. Larry and W. Wayne, "Groop: An Object Oriented Toolkit for Animated 3D Graphics," *Proc. Eighth Ann. Conf. Object-Oriented Programming Systems, Languages, and Applications (OOPSLA '93)*, pp. 309-325, 1993.

[47] *Communicating with Real-Time 3D across Applications, Networks, and Xml Web Services*, Web3D-Consortium, 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," *Proc. ACM Symp. Software Visualization (SoftVis '05)*, pp. 173-182, 2005.

[50] J.F. Hopkins and P.A. Fishwick, "The Rube Framework for Personalized 3-D Software Visualization," *Revised Lectures on Software Visualization, Int'l Seminar*, pp. 368-380, 2002.

[51] J. Heer, S.K. Card, and J.A. Landay, "Prefuse: A Toolkit for Interactive Information Visualization," *Proc. ACM SIGCHI Conf. Human Factors in Computing Systems (CHI '05)*, pp. 421-430, 2005.

[52] A. Telea and L. Voinea, "A Framework for Interactive Visualization of Component-Based Software," *Proc. 30th EUROMICRO Conf. (EUROMICRO '04)*, pp. 567-574, 2004.

[53] K.P. Herndon, A. van Dam, and M. Gleicher, "The Challenges of 3D Interaction: A Chi '94 Workshop," *ACM SIGCHI Bull.*, vol. 26, no. 4, pp. 36-43, 1994.

[54] D.A. Bowman, E. Kruijff, J.J. Laviola 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," *Proc. IEEE Symp. Information Visualization (InfoVis '02)*, p. 57, 2002.

[56] A. Cockburn and B. McKenzie, "Evaluating the Effectiveness of Spatial Memory in 2D and 3D Physical and Virtual Environments," *Proc. ACM SIGCHI Conf. Human Factors in Computing Systems (CHI '02)*, pp. 203-210, 2002.

[57] A.J. Hanson, E.A. Wernert, and S.B. Hughes, "Constrained Navigation Environments," *Proc. Scientific Visualization Conf. (dagstuhl '97)*, p. 95, 1997.

[58] D.A. Bowman and L.F. Hodges, "User Interface Constraints for Immersive Virtual Environment Applications," Technical Report 95-26, Graphics, Visualisation and Usability Center, Georgia Inst. of Technology, 1995.

[59] S. Hughes and M. Lewis, "Robotic Camera Control for Remote Exploration," *Proc. ACM SIGCHI Conf. Human Factors in Computing Systems (CHI '04)*, pp. 511-517, 2004.

[60] A. Ahmed and P. Eades, "Automatic Camera Path Generation for Graph Navigation in 3D," *Proc. Asia-Pacific Symp. Information Visualisation (APVis '05)*, pp. 27-32, 2005.

Authorized licensed use limited to: UNIVERSITY OF WINDSOR. Downloaded on January 15, 2010 at 09:59 from IEEE Xplore. Restrictions apply.

[61] R.P. Darken and J.L. Sibert, "A Toolset for Navigation in Virtual Environments," *Proc. Sixth Ann. ACM Symp. User Interface Software and Technology (UIST '93),* pp. 157-165, 1993.

[62] R. Ingram and S. Benford, "Legibility Enhancement for Information Visualisation," *Proc. IEEE Conf. Visualization,* G.M. Nielson and D. Silver, eds., pp. 209-216, Oct. 1995.

[63] N.G. Vinson, "Design Guidelines for Landmarks to Support Navigation in Virtual Environments," *Proc. ACM SIGCHI Conf. Human Factors in Computing Systems (CHI '99),* pp. 278-285, 1999.

[64] M.C. Chuah, S.F. Roth, J. Mattis, and J. Kolojejchick, "SDM: Selective Dynamic Manipulation of Visualizations," *Proc. Eighth ACM Symp. User Interface Software and Technology (UIST '95),* ser. 3D User Interfaces, pp. 61-70, 1995.

[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?" *Proc. ACM SIGCHI Conf. Human Factors in Computing Systems (CHI '95),* ser. Panels, vol. 2, pp. 174-175, 1995.

[66] R. Koschke, "Software Visualization for Reverse Engineering," *Revised Lectures on Software Visualization, Int'l Seminar,* S. Diehl, ed., 2002.

[67] *Revised Lectures on Software Visualization, Int'l Seminar.* S. Diehl, ed., Springer, 2002.

[68] G. Parker, G. Franck, and C. Ware, "Visualization of Large Nested Graphs in 3D: Navigation and Interaction," *J. Visual Languages and Computing,* vol. 9, no. 3, pp. 299-317, 1998.

[69] J.L. Bentley and B.W. Kernighan, "System for Algorithm Animation," *Computer Systems,* 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," *J. Visual Languages and Computing,* vol. 3, pp. 161-193, 1992.

[71] M.H. Brown and M.A. Najork, "Algorithm Animation Using 3D Interactive Graphics," *Proc. Sixth Ann. ACM Symp. User Interface Software and Technology (UIST '93),* pp. 93-100, 1993.

[72] H. Koike, T. Takada, and T. Masui, "Visualinda: A Framework for Visualizing Parallel Linda Programs," *Proc. IEEE Symp. Visual Languages (VL '97),* p. 174, 1997.

[73] N. Osawa, K. Asai, M. Suzuki, Y. Sugimoto, and F. Saito, "An Immersive Programming System: Ougi," *Proc. 12th Int'l Conf. Artificial Reality and Telexistence (ICAT '02),* pp. 36-43, 2002.

[74] A.R. Teyseyre and M. Campo, "Early Requirements Validation with 3D Worlds," *Computer Systems Science and Eng.,* special issue: automated tools for requirements engineering, no. 20, pp. 61-72, Jan. 2005.

[75] T.A. Alspaugh, B. Tomlinson, and E. Baumer, "Using Social Agents to Visualize Software Scenarios," *Proc. ACM Symp. Software Visualization (SoftVis '06),* pp. 87-94, 2006.

[76] H. Giese, E. Kindler, F. Klein, and R. Wagner, "Reconciling Scenario-Centered Controller Design with State-Based System Models," *Proc. Fourth Int'l Workshop Scenarios and State Machines: Models, Algorithms and Tools (SCESM '05),* pp. 1-5, 2005.

[77] H. Koike and H.-C. Chu, "Vrcs: Integrating Version Control and Module Management Using Interactive 3D Graphics," *Proc. IEEE Symp. Visual Languages (VL '97),* p. 168, 1997.

[78] X. Xie, D. Poshyvanyk, and A. Marcus, "Visualization of CVS Repository Information," *Proc. 13th Working Conf. Reverse Eng. (WCRE '06),* pp. 231-242, 2006.

[79] Y. Hideshima and H. Koike, "Starmine: A Visualization System for Cyber Attacks," *Proc. Asia-Pacific Symp. Information Visualisation (APVis '06),* pp. 131-138, 2006.

[80] A. Bosca, D. Bonino, M. Comerio, S. Grega, and F. Corno, "A Reusable 3D Visualization Component for the Semantic Web," *Proc. 12th Int'l Conf. 3D Web Technology (Web3D '07),* pp. 89-96, 2007.

[81] Y. Frishman and A. Tal, "Visualization of Mobile Object Environments," *Proc. ACM Symp. Software Visualization (SoftVis '05),* pp. 145-154, 2005.

[82] J.I. Maletic, A. Marcus, and L. Feng, "Source Viewer 3D (sv3d): A Framework for Software Visualization," *Proc. 25th Int'l Conf. Software Eng. (ICSE '03),* pp. 812-813, 2003.

[83] D. Bonyuet, M. Ma, and K. Jaffrey, "3D Visualization for Software Development," *Proc. IEEE Int'l Conf. Web Services (ICWS '04),* p. 708, 2004.

[84] P. McIntosh, M. Hamilton, and R.G. van Schyndel, "X3D-UML: Enabling Advanced UML Visualisation through X3D," *Proc. 10th Int'l Conf. 3D Web Technology (Web3D '05),* N.W. John, S. Ressler, L. Chittaro, and D.A. Duce, eds., pp. 135-142, 2005.

[85] M. Balzer and O. Deussen, "Hierarchy Based 3D Visualization of Large Software Structures," *Proc. IEEE Conf. Visualization (VIS '04),* p. 598.4, 2004.

[86] O. Greevy, M. Lanza, and C. Wysseier, "Visualizing Live Software Systems in 3D," *Proc. ACM Symp. Software Visualization (SoftVis '06),* pp. 47-56, 2006.

[87] O. Radfelder and M. Gogolla, "On Better Understanding UML Diagrams through Interactive Three-Dimensional Visualization and Animation," *Proc. Working Conf. Advanced Visual Interfaces (AVI '00),* pp. 292-295, 2000.

[88] G. Langelier, H. Sahraoui, and P. Poulin, "Visualization-Based Analysis of Quality for Large-Scale Software Systems," *Proc. 20th IEEE/ACM Int'l Conf. Automated Software Eng. (ASE '05),* pp. 214-223, 2005.

[89] J.-Y. Vion-Dury and M. Santana, "Virtual Images: Interactive Visualization of Distributed Object-Oriented Systems," *Proc. Ninth Ann. Conf. Object-Oriented Programming Systems, Language, and Applications (OOPSLA '94),* pp. 65-84, 1994.

[90] L. Feijs and R.D. Jong, "3D Visualization of Software Architectures," *Comm. 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," *Proc. Ninth IEEE Int'l Conf. Eng. of Computer-Based Systems (ECBS '02),* pp. 139-146, 2002.

[92] J. Rekimoto and M. Green, "The Information Cube: Using Transparency in 3D Information Visualization," *Proc. Third Ann. Workshop Information Technologies and Systems,* pp. 125-132, 1993.

[93] T. Panas, R. Berrigan, and J. Grundy, "A 3D Metaphor for Software Production Visualization," *Proc. Seventh Int'l Conf. Information Visualisation (IV '03),* p. 314, 2003.

[94] S.M. Charters, C. Knight, N. Thomas, and M. Munro, "Visualisation for Informed Decision Making; from Code to Components," *Proc. 14th Int'l Conf. Software Eng. and Knowledge Eng. (SEKE '02),* pp. 765-772, 2002.

[95] C. Knight and M. Munro, "Virtual but Visible Software," *Proc. Fourth Int'l Conf. Information Visualisation (IV '00),* p. 198, 2000.

[96] D. Ploix, "Building Program Metaphors," *Proc. Eighth Ann. PPIG Workshop,* pp. 125-129, 1996.

[97] H. Graham, H.Y. Yang, and R. Berrigan, "A Solar System Metaphor for 3D Visualisation of Object Oriented Software Metrics," *Proc. Australasian Symp. Information Visualisation,* pp. 53-59, 2004.

[98] B.A. Malloy and J.F. Power, "Using a Molecular Metaphor to Facilitate Comprehension of 3D Object Diagrams," *Proc. IEEE Symp. Visual Languages and Human-Centric Computing (VL/HCC '05),* pp. 233-240, 2005.

[99] K. Kahn, "Drawings on Napkins, Video-Game Animation, and Other Ways to Program Computers," *Comm. 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," *Proc. Sixth ACM SIGCHI New Zealand Chapter's Int'l Conf. Computer-Human Interaction (CHINZ '05),* pp. 53-60, 2005.

[101] J. Mackinlay, "Automating the Design of Graphical Presentations of Relational Information," *ACM Trans. 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.* Prentice Hall PTR, 1998.

[103] I. Herman, G. Melançon, and M.S. Marshall, "Graph Visualization and Navigation in Information Visualization: A Survey," *IEEE Trans. Visualization and Computer Graphics,* vol. 6, no. 1, pp. 24-43, Jan.-Mar. 2000.

[104] H.-J. Schulz and H. Schumann, "Visualizing Graphs—A Generalized View," *Proc. 10th Int'l Conf. Information Visualization (IV '06),* pp. 166-173, 2006.

[105] C. Ware and P. Mitchell, "Visualizing Graphs in Three Dimensions," *ACM Trans. Applied Perception,* vol. 5, no. 1, pp. 1-15, 2008.

[106] C. Ware and P. Mitchell, "Reevaluating Stereo and Motion Cues for Visualizing Graphs in Three Dimensions," *Proc. Second Symp. Applied Perception in Graphics and Visualization (APGV '05),* pp. 51-58, 2005.

[107] B. Shneiderman and A. Aris, "Network Visualization by Semantic Substrates," *IEEE Trans. Visualization and Computer Graphics,* vol. 12, no. 5, pp. 733-740, Sept./Oct. 2006.

Authorized licensed use limited to: UNIVERSITY OF WINDSOR. Downloaded on January 15, 2010 at 09:59 from IEEE Xplore. Restrictions apply.

[108] M.-A. Storey, C. Best, and J. Michaud, "Shrimp Views: An Interactive Environment for Exploring Java Programs," *Proc. Ninth Int'l Workshop Program Comprehension (IWPC '01),* p. 111, 2001.

[109] M.-A. Storey, N.F. Noy, M. Musen, C. Best, R. Fergerson, and N. Ernst, "Jambalaya: An Interactive Environment for Exploring Ontologies," *Proc. Seventh Int'l Conf. Intelligent User Interfaces (IUI '02),* p. 239, 2002.

[110] J.I. Maletic, A. Marcus, G. Dunlap, and J. Leigh, "Visualizing Object-Oriented Software in Virtual Reality," *Proc. Ninth Int'l Workshop Program Comprehension (IWPC '01),* p. 26, 2001.

[111] S.P. Reiss, "Bee/Hive: A Software Visualization Back End," *Proc. Workshop Software Visualizations (ICSE '01),* pp. 44-48, 2001.

[112] T. Munzner, "H3: Laying Out Large Directed Graphs in 3D Hyperbolic Space," *Proc. IEEE Symp. Information Visualization (InfoVis '97),* p. 2, 1997.

[113] C. Lewerentz and A. Noack, "Crococosmos—3D Visualization of Large Object-Oriented Programs," *Graph Drawing Software,* pp. 279-297, Springer, 2003.

[114] T. Dwyer, "Three Dimensional Uml Using Force Directed Layout," *Proc. Asia-Pacific Symp. Information Visualisation (APVis '01),* pp. 77-85, 2001.

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

[116] M. Campo, T. Price, and A. Teyseyre, "Uma Abordagem 3D Para A Visualização de Padrões de Projeto," *Proc. Anais XI Simpósio Brasileiro de Engenharia de Software,* Oct. 1997.

[117] G.G. Robertson, J.D. Mackinlay, and S.K. Card, "Cone Trees: Animated 3D Visualizations of Hierarchical Information," *Proc. ACM SIGCHI Conf. Human Factors in Computing Systems (CHI '91),* pp. 189-194, 1991.

[118] R. Berghammer and A. Fronk, "Applying Relational Algebra in 3D Graphical Software Design," *Proc. Seventh Int'l Seminar on Relational Methods in Computer Science (RelMiCS '03),* R. Berghammer, B. Möller, and G. Struth, eds., pp. 62-74, 2003.

[119] W. Wang, H. Wang, G. Dai, and H. Wang, "Visualization of Large Hierarchical Data by Circle Packing," *Proc. ACM SIGCHI Conf. Human Factors in Computing Systems (CHI '06),* pp. 517-520, 2006.

[120] F. van Ham and J.J. van Wijk, "Beamtrees: Compact Visualization of Large Hierarchies," *Proc. IEEE Symp. Information Visualization (InfoVis '02),* p. 93, 2002.

[121] B. Johnson and B. Shneiderman, "Tree-Maps: A Space-Filling Approach to the Visualization of Hierarchical Information Structures," *Proc. Second IEEE Conf. Visualization,* pp. 284-291, 1991.

[122] S.P. Reiss, "An Overview of Bloom," *Proc. ACM SIGPLAN-SIGSOFT Workshop Program Analysis for Software Tools and Eng. (PASTE '01),* pp. 2-5, 2001.

[123] S.G. Eick, J.L. Steffen, J. Eric, and E. Sumner, "Seesoft—A Tool for Visualizing Line Oriented Software Statistics," *IEEE Trans. Software Eng.,* vol. 18, no. 11, pp. 957-968, Nov. 1992.

[124] P. Young, "Visualising Software in Cyberspace," PhD dissertation, Univ. 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," *Proc. 12th IEEE Int'l Conf. Eng. Complex Computer Systems (ICECCS '07),* pp. 217-228, 2007.

[126] R. Wettel and M. Lanza, "Program Comprehension through Software Habitability," *Proc. 15th IEEE Int'l Conf. Program Comprehension (ICPC '07),* pp. 231-240, 2007.

[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.* pp. 442-450, Morgan Kaufmann, 1999.

[128] A. Celentano, M. Nodari, and F. Pittarello, "Adaptive Interaction in Web3d Virtual Worlds," *Proc. Ninth Int'l Conf. 3D Web Technology (Web3D '04),* pp. 41-50, 2004.

[129] U. Wiss and D.A. Carr, "An Empirical Study of Task Support in 3D Information Visualizations," *Proc. Int'l Conf. Information Visualisation (IV '99),* p. 392, 1999.

[130] R. Dachselt and M. Hinz, "Three-Dimensional Widgets Revisited-Towards Future Standardization," *Proc. IEEE VR Workshop New Directions in 3D User Interfaces,* D. Bowman, B. Froehlich, Y. Kitamura, and W. Stuerzlinger, eds., pp. 89-92, 2005.

[131] R. Dachselt and A. Hübner, "Virtual Environments: Three-Dimensional Menus: A Survey and Taxonomy," *Computer Graphics,* 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," *Proc. Fifth Int'l Conf. Information Visualisation (IV '01),* p. 517, 2001.

[133] M. Tory and T. Moller, "Evaluating Visualizations: Do Expert Reviews Work?" *IEEE Computer Graphics and Applications,* vol. 25, no. 5, pp. 8-11, Sept./Oct. 2005.

[134] A. Kerren, A. Ebert, and J. Meyer, "Introduction to Human-Centered Visualization Environments," *Human-Centered Visualization Environments, Gi-Dagstuhl Research Seminar,* A. Kerren, A. Ebert, and J. Meyer, eds., chapter 1, pp. 1-9, Springer, 2007.

[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, July/Aug. 2003.

[136] P. Rheingans, "Are We There Yet? Exploring with Dynamic Visualization," *IEEE Computer Graphics and Applications,* vol. 22, no. 1, pp. 6-10, Jan./Feb. 2002.

[137] M. Winckler, P. Palanque, and C. Freitas, "Tasks and Scenario-Based Evaluation of Information Visualization Techniques," *Proc. Third Ann. Conf. Task Models and Diagrams (TAMODIA '04),* pp. 165-172, 2004.

[138] D. House, V. Interrante, D. Laidlaw, R. Taylor, and C. Ware, "Design and Evaluation in Visualization Research," *Proc. IEEE Conf. Visualization,* p. 117, 2005.

[139] C. Plaisant, "The Challenge of Information Visualization Evaluation," *Proc. Working Conf. Advanced Visual Interfaces (AVI '04),* pp. 109-116, 2004.

[140] G. Ellis and A. Dix, "An Explorative Analysis of User Evaluation Studies in Information Visualisation," *Proc. AVI Workshop Beyond Time and Errors (BELIV '06),* pp. 1-7, 2006.

[141] C. Chen and Y. Yu, "Empirical Studies of Information Visualization: A Meta-Analysis," *Int'l J. Human-Computer Studies,* vol. 53, no. 5, pp. 851-866, 2000.

[142] O. Kulyk, R. Kosara, J. Urquiza, and I. Wassink, "Human-Centered Aspects," *Human-Centered Visualization Environments, Gi-Dagstuhl Research Seminar,* A. Kerren, A. Ebert, and J. Meyer, eds., chapter 2, pp. 10-75, 2007.

[143] J. Stasko, A. Badre, and C. Lewis, "Do Algorithm Animations Assist Learning?: An Empirical Study and Analysis," *Proc. Confs. Human Factors in Computing Systems (INTERACT '93 and CHI '93),* pp. 61-66, 1993.

[144] D. Sun and K. Wong, "On Evaluating the Layout of Uml Class Diagrams for Program Comprehension," *Proc. 13th Int'l Workshop Program Comprehension (IWPC '05),* 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," *Proc. Fourth Int'l Conf. Cognitive Technology (CT '01),* pp. 325-341, 2001.

[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," *Proc. ACM Symp. Software Visualization (SoftVis '05),* pp. 193-202, 2005.

[147] H.C. Purchase, L. Colpoys, M. McGill, and D. Carrington, "Uml Collaboration Diagram Syntax: An Empirical Study of Comprehension," *Proc. First Workshop Visualizing Software for Understanding and Analysis (VISSOFT '02),* 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," *Proc. 15th IEEE Int'l Conf. Program Comprehension (ICPC '07),* pp. 221-230, 2007.

[149] S. Yusuf, H. Kagdi, and J.I. Maletic, "Assessing the Comprehension of Uml Class Diagrams via Eye Tracking," *Proc. 15th IEEE Int'l Conf. Program Comprehension (ICPC '07),* pp. 113-122, 2007.

[150] Y.-G. Guéhéneuc, "Taupe: Towards Understanding Program Comprehension," *Proc. Conf. Center for Advanced Studies on Collaborative Research (CASCON '06),* p. 1, 2006.

[151] P. Irani and C. Ware, "The Effect of a Perceptual Syntax on the Learnability of Novel Concepts," *Proc. Eighth Int'l Conf. Information Visualisation (IV '04),* pp. 308-314, 2004.

Authorized licensed use limited to: UNIVERSITY OF WINDSOR. Downloaded on January 15, 2010 at 09:59 from IEEE Xplore. Restrictions apply.

[152] A. Marcus, D. Comorski, and A. Sergeyev, "Supporting the Evolution of a Software Visualization Tool through Usability Studies," *Proc. 13th Int'l Workshop Program Comprehension (IWPC '05),* pp. 307-316, 2005.

[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," *Proc. Third Working Conf. Reverse Eng. (WCRE '96),* p. 31, 1996.

[154] J.A. Jones and M.J. Harrold, "Empirical Evaluation of the Tarantula Automatic Fault-Localization Technique," *Proc. 20th IEEE/ACM Int'l Conf. Automated Software Eng. (ASE '05),* ACM Press, pp. 273-282, 2005.

[155] B. de Alwis, G.C. Murphy, and M.P. Robillard, "A Comparative Study of Three Program Exploration Tools," *Proc. 15th IEEE Int'l Conf. Program Comprehension (ICPC '07),* pp. 103-112, 2007.

[156] S. Bassil and R.K. Keller, "Software Visualization Tools: Survey and Analysis," *Proc. Ninth Int'l Workshop Program Comprehension (IWPC '01),* pp. 7-17, 2001.

[157] R.A. Amar and J.T. Stasko, "Knowledge Precepts for Design and Evaluation of Information Visualizations," *IEEE Trans. Visualization and Computer Graphics,* vol. 11, no. 4, pp. 432-442, July-Sept. 2005.

[158] C. North, "Toward Measuring Visualization Insight," *IEEE Computer Graphics and Applications,* vol. 26, no. 3, pp. 6-9, May/June 2006.

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

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

[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," *Proc. IEEE Virtual Reality Conf. (VR '02),* p. 183, 2002.

[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," *Proc. IEEE Virtual Reality Conf. (VR '01),* p. 89, 2001.

[163] X. Zhang and D. Gračanin, "From Coarse-Grained Components to Dve Applications: A Service- and Component-Based Framework," *Proc. 12th Int'l Conf. 3D Web Technology (Web3D '07),* pp. 113-121, 2007.

[164] J.-D. Fekete, "The Infovis Toolkit," *Proc. IEEE Symp. Information Visualization (InfoVis '04),* pp. 167-174, 2004.

[165] R. Orosco and M. Campo, "Mamp: A Design Model for Object-Oriented Visualization Systems," *Eng. for Human Computer Interaction,* C.S. and D.P., eds., Kluwer Academic Publishers, 1998.

[166] M. Campo, R. Orosco, and A. Teyseyre, "Automatic Abstraction Management in Information Visualization Systems," *Proc. IEEE Conf. Information Visualisation (IV '97),* p. 50, 1997.

[167] M. Meyer, T. Gîrba, and M. Lungu, "Mondrian: An Agile Information Visualization Framework," *Proc. ACM Symp. Software Visualization (SoftVis '06),* pp. 135-144, 2006.

[168] Q. Wang, W. Wang, R. Brown, K. Driesen, B. Dufour, L. Hendren, and C. Verbrugge, "Evolve: An Open Extensible Software Visualization Framework," *Proc. ACM Symp. Software Visualization (SoftVis '03),* p. 37-ff, 2003.

[169] A. Fronk and J. Schröder, "Effects—A Framework for 3D Software Visualization," *Poster IEEE Int'l Conf. Automated Software Eng. (ASE '04),* Sept. 2004.

[170] M. Campo and T. Price, "Luthier—Building Framework-Visualization Tools," *Implementing Object-Oriented Application Frameworks: Frameworks at Work,* M. Fayad and R. Johnson, eds., John Wiley & Sons, 1999.

[171] W. Löwe and T. Panas, "Rapid Construction of Software Comprehension Tools," *Int'l J. Software Eng. and Knowledge Eng.,* vol. 15, no. 6, pp. 905-1023, Dec. 2005.

[172] R.I. Bull, M.-A. Storey, J.-M. Favre, and M. Litoiu, "An Architecture to Support Model Driven Software Visualization," *Proc. 14th IEEE Int'l Conf. Program Comprehension (ICPC '06),* pp. 100-106, 2006.

[173] M. Itoh and Y. Tanaka, "3D Component-Based Visualization Framework for Generating Simple 3D Applications Using Web Services," *Proc. IEEE/WIC/ACM Int'l Conf. Web Intelligence (WI '06),* pp. 823-830, 2006.

[174] Y. Okada and Y. Tanaka, "Intelligentbox: A Constructive Visual Software Development System for Interactive 3D Graphic Applications," *Proc. Computer Animation (CA '95),* p. 114, 1995.

[175] R.I. Bull, C. Best, and M.-A. Storey, "Advanced Widgets for Eclipse," *Proc. OOPSLA Workshop Eclipse Technology eXchange (Eclipse '04),* pp. 6-11, 2004.

[176] A. Fronk, A. Bruckhoff, and M. Kern, "3D Visualisation of Code Structures in Java Software Systems," *Proc. ACM Symp. Software Visualization (SoftVis '06),* pp. 145-146, 2006.

[177] A. Tal and D. Dobkin, "Visualization of Geometric Algorithms," *IEEE Trans. Visualization and Computer Graphics,* vol. 1, no. 2, pp. 194-204, June 1995.

[178] M. Najork, "Web-Based Algorithm Animation," *Proc. 38th Conf. Design Automation (DAC '01),* pp. 506-511, 2001.

[179] H. Koike and H.-C. Chu, "How Does 3D Visualization Work in Software Engineering?: Empirical Study of a 3D Version/Module Visualization System," *Proc. 20th Int'l Conf. Software Eng. (ICSE '98),* pp. 516-519, 1998.

[180] M. Burch, S. Diehl, and P. Weibgerber, "Visual Data Mining in Software Archives," *Proc. ACM Symp. Software Visualization (SoftVis '05),* pp. 37-46, 2005.

[181] O. Greevy, M. Lanza, and C. Wysseier, "Visualizing Feature Interaction in 3-D," *Proc. Third IEEE Int'l Workshop Visualizing Software for Understanding and Analysis (VISSOFT '05),* p. 30, 2005.

[182] E. Kindler and C. Páles, "3D-Visualization of Petri Net Models: Concept and Realization," *Proc. 25th Int'l Conf. Applications and Theory of Petri Nets (Petri Nets '04),* J. Cortadella and W. Reisig, eds., pp. 464-473, 2004.

[183] A. Bosca and D. Bonino, "Ontosphere3d: A Multidimensional Visualization Tool for Ontologies," *Proc. 17th Int'l Conf. Database and Expert Systems Applications (DEXA '06),* pp. 339-343, 2006.

[184] A. Kerren and J.T. Stasko, "Algorithm Animation—Introduction," *Revised Lectures on Software Visualization, Int'l Seminar,* pp. 1-15, Springer, 2002.

[185] M.H. Brown and M. Najork, "Algorithm Animation Using Interactive 3D Graphics," *Software Visualization: Programming as a Multimedia Experience,* J. Stasko, J. Domingue, M.H. Brown, and B.A. Price, eds., chapter 9, pp. 119-135, MIT Press, 1997.

[186] K. Dunsire, T. O'Neill, M. Denford, and J. Leaney, "The Abacus Architectural Approach to Computer-Based System and Enterprise Evolution," *Proc. 12th IEEE Int'l Conf. and Workshops on the Eng. of Computer-Based Systems (ECBS '05),* pp. 62-69, 2005.

[187] M. Tory and T. Moller, "Human Factors in Visualization Research," *IEEE Trans. Visualization and Computer Graphics,* vol. 10, no. 1, pp. 72-84, Jan.-Mar. 2004.

[188] B. Sengupta, S. Chandra, and V. Sinha, "A Research Agenda for Distributed Software Development," *Proc. 28th Int'l Conf. Software Eng. (ICSE '06),* pp. 731-740, 2006.

[189] J. Wood, H. Wright, and K. Brodlie, "Collaborative Visualization," *Proc. Eighth IEEE Conf. Visualization,* p. 253-ff, 1997.

[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 Computer Graphics and Applications,* vol. 25, no. 4, pp. 24-33, July/Aug. 2005.

[192] L. Chittaro, "Visualizing Information on Mobile Devices," *Computer,* vol. 39, no. 3, pp. 40-45, Mar. 2006.

[193] P. Baudisch, "Interacting with Large Displays," *Computer,* vol. 39, no. 4, pp. 96-97, Apr. 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, July/Aug. 2005.

Authorized licensed use limited to: UNIVERSITY OF WINDSOR. Downloaded on January 15, 2010 at 09:59 from IEEE Xplore. Restrictions apply.

Alfredo R. Teyseyre received the BSc degree in systems engineering and the master's degree in systems engineering from the Universidad Nacional del Centro de la Provincia de Buenos Aires (UNICEN), Tandil, Argentina, in 1997 and 2001, respectively. Currently, he is a PhD candidate in the Department of Computer Science and ISISTAN Research Institute, Facultad de Ciencias Exactas, UNICEN. He is also an adjunct professor in the Department of Computer Science, UNICEN. His research interests include software visualization, information visualization, software architecture and frameworks, and lightweight formal methods.

Marcelo R. Campo received the systems engineer degree from the Universidad Nacional del Centro de la Provincia de Buenos Aires (UNICEN), Tandil, Argentina in 1988 and the PhD degree in computer science from the Universidade Federal do Rio Grande do Sul, Porto Alegre, Brazil, in 1997. Currently, he is an associate professor in the Computer Science Department and the head of the ISISTAN Research Institute, Facultad de Ciencias Exactas, UNICEN. He is also a research fellow of the National Council for Scientific and Technical Research of Argentina (CONICET), Buenos Aires, Argentina, and the national coordinator of the IT area. He has more than 50 papers published in main conference proceedings and journals about software engineering topics. His research interests include intelligent aided software engineering, software architecture and frameworks, agent technology, and software visualization. He is a member of the IEEE.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.

Authorized licensed use limited to: UNIVERSITY OF WINDSOR. Downloaded on January 15, 2010 at 09:59 from IEEE Xplore. Restrictions apply.