A Meta-Study of Software Visualization Effectiveness

CHRISTOPHER D. HUNDHAUSEN
Department of Computer and Information Science, University of Oregon, Eugene, OR, 97403–1202
March 3, 1996

The overriding goal of software visualization (SV) technology is to be *effective*—that is, to provide functionality that people actually want to use (*usefulness*), and to provide non-problematic access to that functionality (*usability*). Nonetheless, the eight extant taxonomies of SV focus almost exclusively on technology *expressiveness*: the kinds of visualizations that can be produced with a system, as well as the methods prescribed by the system for generating and interacting with those visualizations. However, while expressiveness is *necessary* for effectiveness, it certainly is not *sufficient*. To enhance the navigational markings of the expressiveness-laden map charted by past SV taxonomies, this meta-study critically assesses the state-of-the-art with respect to SV effectiveness. The meta-study's focus on effectiveness implies the need to abandon both the SV system as its unit of analysis, and the SV systems literature as its principal data. Accordingly, SV is reconceptualized as a collection of human tasks; the empirical studies that have considered those tasks form the study's data. The meta-study's data differ along three key dimensions: their *theories of effectiveness*, their *research techniques*, and their *research foci*. Different choices along any of these dimensions lead to accounts of effectiveness that differ markedly—both qualitatively and quantitatively. To draw out the differences to which alternative choices lead, the meta-study pursues in-depth analyses of *divisions of labor* (indicating theories of effectiveness); *units of analysis*, *data collection/analysis methods*, and *desired results* (indicating research techniques); and *research questions*, *SV artifacts*, *target programs*, *participants*, and *SV tasks* (research foci). In light of the SV effectiveness studies' overriding interest in pertaining to *SV in practice*, a provisional analysis of their *ecological validity* is additionally offered. A synthesis of the analysis both proposes theories of effectiveness, research techniques, and research foci that appear ripe for future research, and recommends that future research be firmly grounded in ethnographic descriptions of SV in practice.

1. Introduction

SOFTWARE VISUALIZATION (SV), which purports to foster understanding and efficient use of computer software by representing it graphically (Price, Baecker, & Small, 1993), has garnered widespread interest over the past two decades. Since its genesis in the early 1980s, SV technology has been used, for example,

- to track down bugs in logic programs (e.g., Price, 1990);
- to learn about the basic operations of an abstract data type in a computer science laboratory (e.g., Naps, 1990);
- to find performance bottlenecks in a parallel program (e.g., Heath & Etheridge, 1991);
- to monitor and steer the execution of long-running, resource-intensive scientific simulations (e.g., Gu *et al.*, 1994); and
- to give lectures on sorting algorithms in electronic classrooms (e.g., Brown, 1988a).

Observe that, while the applications of SV have been diverse, they all have at least one thing in common: namely, in all cases, humans enlist SV technology to *assist them in these tasks*.

1.1 *Orphan systems* and *system roulette*

Given that the overriding purpose of visualization is to benefit humans, it is noteworthy that relatively few SV systems are actually in use today. Based on their extensive survey of SV systems, Price, Baecker, and Small (1993) conclude that

> [o]ver one hundred software visualization prototypes have been built in the last twenty years. . .The number that have seen any kind of production use, particularly in the domain of tools for professional programmers, is particularly small. (p. 261)

Why are there so many *orphan SV systems*—that is, systems abandoned by their creators shortly after the journal or proceedings paper was written and accepted? While that question is difficult to address in the general case, a survey of the dominant themes in SV research indicates that a development strategy I call *system roulette* may be to blame. In system roulette, technical challenges, as well as a desire for technological innovation, are the principal forces behind system design. As a result, researchers fail to obtain a clear idea of what their systems can and should be used for. That failure, in turn, leads to systems that either (a) do not adequately address the needs of their users, or (b) are simply not useful in solving the problem they purportedly address.

The results of an empirical study of the LENS SV system (Mukherjea & Stasko, 1994) I conducted (Hundhausen, 1993) well illustrate the system roulette phenomenon. Mukherjea and Stasko (1994) tout LENS as a "visual debugging system" whose "application-specific views could be a valuable debugging aid" (p. 215), because they "present [a] program in the way that the programmer conceptualizes it" (p. 217). However, my study found that, when given the choice between (a) *textual debugging*, with its concomitant guarantee that all bugs are in the program itself, and (b) *LENS debugging*, which requires one first to engage in additional programming in order to map a buggy program to an animation, programmers will choose the former. Notice that in the latter, the process of mapping a buggy program to an animation introduces an extra layer for potential bugs. Thus, assuming that programmers will be able to recognize a bug visually (which was not at all clear from the study), they may not be able to determine whether that bug is in the underlying program, or in the mapping that produced the animation. Since my participants did not want to deal with this added uncertainty, they avoided the use of LENS altogether for their debugging tasks.

We see, then, that while LENS may well be useful in some other endeavor[1], the study's results suggest that it may be ill-suited to debugging. In other words, despite its *prima facie* promise, Mukherjea and Stasko's vision of high-level visual debugging appears to have given rise to system roulette.

1.2 The role of taxonomies in encouraging *system roulette*

In order to understand what might have compelled Mukherjea and Stasko to build a visual debugging system, let us consider the role of *taxonomies* in a technological field such as SV. As Price, Baecker, and Small (1993) point out, taxonomies enable researchers "to discuss the merits of existing systems, classify new ones (to see if they really are new), and identify gaps which suggest promising areas for further development" (p. 330). In other words, by defining system spaces, and by mapping extant systems onto those spaces, taxonomies define the very notion of a *contribution* within a technological field.

A survey of the eight extant taxonomies of SV technology (see Table 1) reveals that system *expressiveness* has served as the main organizing principle. In particular, the taxonomies have been concerned with three main questions: (1) What kinds of programs can be visualized with a given SV system?, (2) What kinds of visualizations can an SV system produce?, and (3) What methods can one use to produce and interact with them?

Given the focus of past taxonomies on expressiveness, it should come as no surprise that SV system roulette, as exemplified by Mukherjea and Stasko's LENS system, is so widely practiced. Indeed, in their view, a main selling point of LENS is that it "occupies a unique niche" (Muherjea & Stasko, 1994, p. 215) between the extant SV systems that generate higher-level, application-specific views, and the extant SV systems that generate lower-level, canonical views of program structures. Insofar as they provide such "unique niches" for technological innovation, the extant taxonomies implicitly reward SV systems—like LENS—that "out-express" their predecessors in some way. It follows, then, that expressiveness-centered *taxonomies* provide ample motivation for SV system builders like

[1]To LENS's credit, study participants found it to be useful in quickly producing canonical visualizations of certain sorting algorithms.

Mukherjea and Stasko to make technological challenge and innovation, rather than humans, their central concern—that is, to engage in system roulette.

SV TAXONOMY	DESCRIPTIVE DIMENSIONS
(Myers, 1986, 1990)	*Aspect* (code, data, algorithms) × *Form* (static, animated)
(Shu, 1988)	*What is visualized* (data or information about data, program and/or execution, software design)
(Brown, 1988*b*)	*Content* (direct, synthetic) × *Persistence* (current, history) × *Transformation* (incremental, discrete)
(Stasko & Patterson, 1992)	*Aspect* × *Abstractness* × *Animation* × *Automation*
(Singh & Chignell, 1992)	*What is visualized* (program, algorithm, data) × *Form* (static, dynamic)
(Kraemer & Stasko, 1993)	*Visualization task* (data collection, data analysis, storage, display) × *Visualization purpose* (debugging, performance evaluation or optimization, program visualization)
(Roman & Cox, 1993)	*Scope* × *Abstraction* × *Specification method* × *Interface* × *Presentation*
(Price, Baecker, & Small, 1993)	*Scope* × *Content* × *Form* × *Method* × *Interaction* × *Effectiveness*

Table 1. The descriptive dimensions of the eight extant taxonomies of SV

1.3 From *expressiveness* to *effectiveness*

In light of the potentially deleterious consequences of expressiveness-centric taxonomies, it seems reasonable to seek an alternative characterization of SV technology that places it within the broader context of human use. To that end, Mackinlay (1986) distinguishes two descriptive properties of visualization systems[2]: *expressiveness* and *effectiveness*. As we have seen, system *expressiveness* indicates what kinds of visualizations can be produced with the system, as well as the methods prescribed by the system for generating and interacting with those visualizations. In contrast, system's *effectiveness*, to a first approximation, can be defined as the extent to which the system provides functionality that people actually want to use (*usefulness*), as well as the extent to which people can make use of that functionality nonproblematically (*usability*). From these two definitions, it follows that system expressiveness is necessary, but not sufficient, for system effectiveness.

As Table 1 indicates, not all of the extant taxonomies of SV completely ignore effectiveness. While maintaining the focus of its predecessors, the recent taxonomy of Price Baecker and Small (1993; henceforth *PBS*) momentarily departs from its "principled" black-box model to include the notion of effectiveness (see Figure 1). Given the overall *purpose* of an SV system, PBS define a SV system's effectiveness in terms of its *appropriateness and clarity* (a subjective measure), the extent to which it has undergone *empirical evaluation*, and the extent to which it is actually *used in practice*. PBS proceed to use their definition as a basis for assessing the effectiveness of twelve extant SV systems.

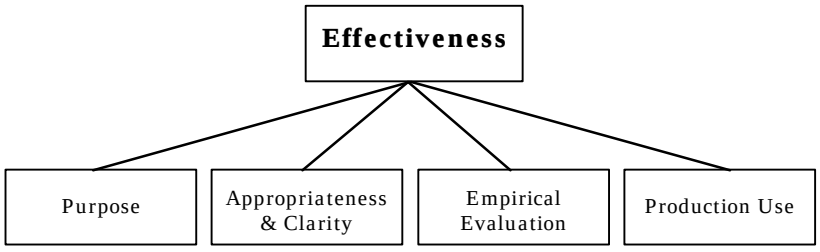

Figure 1. The definition of effectiveness proposed by the taxonomy of Price, Baecker, and Small (1993)

Insofar as it places a notion of effectiveness on the previously expressiveness-laden SV map, the PBS taxonomy takes an important first step toward bringing humans into the SV technology equation. However, the taxonomy's overriding interest in SV *technology* ultimately prevents it from providing

[2]Note that Mackinlay (1986) uses these terms to describe visualization *languages*. Since any SV system implicitly defines a SV language (Douglas, Hundhausen, & McKeown, 1995, 1996), however, the terms apply equally well to SV *systems*.

more than a perfunctory treatment of effectiveness. Indeed, in the end, the taxonomy's assessment of effectiveness rests largely on the subjective judgments of its authors (see p. 245), who are unable to clarify what they mean by "good experimental evaluation" (p. 259) much less to identify and discuss those good experimental evaluation efforts upon which they base their assessments of the 12 individual SV systems.

1.4 The need for complementary studies of SV effectiveness

If SV scholars are to take the overriding goal of SV systems (i.e., *effectiveness*) seriously, it is incumbent upon them to move beyond the expressiveness-centric perspective of past taxonomies for at least two reasons. First, SV system developers, who presumably constitute the main beneficiaries of SV taxonomies, endeavor to build effective systems. How can they learn from past research if the integrative reviews of that research do not provide a principled assessment of the effectiveness of scenarios in which SV technology has been previously enlisted?

Second, and more important, if it is true that system roulette leads to orphan systems, then one way to retard the development of orphan systems—and thereby to accelerate the development of systems that people actually use—is to discourage system roulette. As I have suggested above, taxonomies play an instrumental role in setting the agenda for future research. By broadening the perspective of the extant taxonomies, an integrative review of what is known about SV effectiveness thus holds promise in encouraging the development of effective systems.

1.5 Preview

By critically assessing state of the art with respect to SV effectiveness, this meta-study enhances the navigational markings of the expressiveness-laden map charted by past SV taxonomies. To do this requires that the meta-study abandon both the SV system as its unit of analysis, and the SV systems literature as is principal data. Drawing from a formal definition of SV, Section 2 sets the boundaries for the meta-study by reconceptualizing SV as a *collection of human tasks*. The literature that has considered SV technology in relation to those tasks is thus the central data for the meta-study. In Section 3, I present those data in the form of a comprehensive review of the SV effectiveness literature, organizing my presentation around a taxonomy of research techniques for studying SV tasks. Section 4 presents the meta-study's analysis, which probes three key dimensions along which the SV effectiveness studies differ: *theory of effectiveness*, *research technique*, and *research foci* (research questions, programs, SV systems, people, and tasks). In addition, the analysis assesses the extent to which the SV effectiveness studies apply to SV in practice (i.e., their *ecological validity*). Finally, Section 5 synthesizes the analysis by presenting an agenda for future research into SV effectiveness.

2. Scope: The tasks of SV

The scope of this meta-study must be firmly grounded in both a definition of effectiveness, and a definition of SV. The notion of effectiveness, as it will be used in this meta-study, focuses on the union of humans and technology within the context of a *task*. Associated with such a task is (a) a particular *objective* to be fulfilled, (b) a particular *group of humans* who share that objective, and (c) a particular SV *artifact*[3] that they will enlist; and (d) a particular *target program* to be visualized. Within the context of such a task, effectiveness thus responds to the question, To what extent does the system assist the group in completing the task?

Two points should be made with respect to this definition of effectiveness. First, notice that answers to this question address either or both of the aspects of effectiveness introduced above: *usefulness*

[3]To emphasize that computer-based technology is just one of many alternative means for expressing graphical representations of software, I shall deliberately use the term *artifact*, as opposed to *system*, in my general discussions of SV. The term *artifact* encompasses any technology that can be used in the service of SV, including conventional computer-based systems, as well as "low tech" art supplies such as pens, construction paper, and scissors.

and *usability*. A *useful* system provides the functionality people want in order to complete the task. On the other hand, a *usable* system empowers people to focus on the task itself, without puzzling over how to use the system. Second, and more important, notice that answers to that question may take many forms, both qualitative or quantitative, depending intimately on the task at hand, the people performing the task, the SV system, and other peculiarities of the context. Thus, statements of effectiveness necessarily address not the SV technology alone (as the extant taxonomies of SV do), not humans alone, but rather *the union of the two within the context of the task*.

Given that the notion of effectiveness derives its meaning from the peculiarities of a *task*, as just described, what tasks are within the scope of studies of *SV* effectiveness? While acknowledging that studies of other tasks are plainly relevant to SV effectiveness,[4] I restrict the scope of this meta-study to those tasks that can be derived from an established definition of SV. Roman and Cox's (1993) definition focuses on *a formal model of the process*, which includes three fundamental entities: *program*, *mapping*, and *visualization*. Because of its focus on the *process* by which a program is transformed into a visualization, it provides a crisp delineation of the high-level tasks of SV. Indeed, each stage of the process implies the need for one or more user tasks.

Figure 2 illustrates the correspondence between Roman and Cox's model stages and user tasks. Below, I elaborate further on each of these tasks, drawing on the SV systems literature to illustrate what they might involve. I conclude the section by considering how the tasks relate to one another.

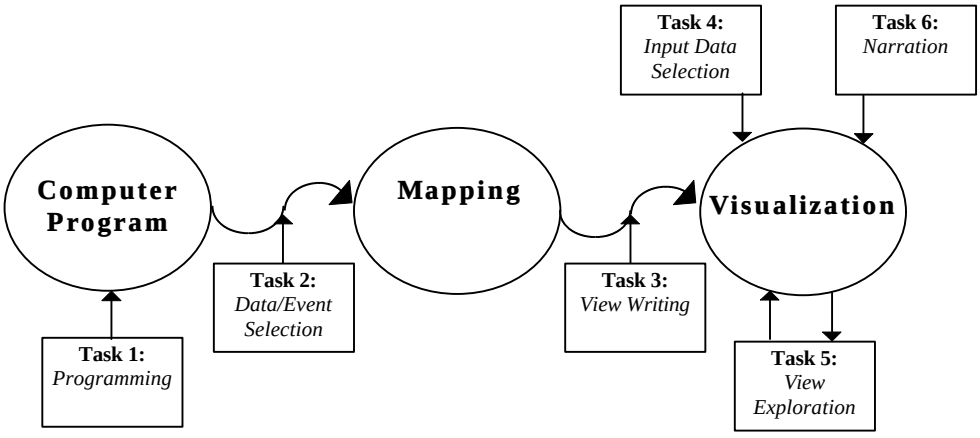

Figure 2. Deriving SV's six high level tasks from Roman and Cox's formal model of SV

2.1 Programming

To produce a computer program to be visualized, one engages in the task of *programming*. While the general tasks of programming have been widely studied (see, e.g., Pennington & Grabowski, 1990), this meta-study restricts its definition of programming tasks to those that involve the selection, creation, or modification of *a program that will ultimately be visualized*.[5]

2.2 Algorithm event and data selection

In the first stage of the mapping process, one engages in *algorithm event and data selection* by selecting program data and events that are of interest to the visualization. For example, under

[4]See, e.g., (Rieber, Boyce, & Assad, 1900; Rieber, 1990; Mayer & Anderson, 1991; Palmiteer & Elkerton, 1993; Pane, Corbett, & John, 1996) for studies of learning and comprehension tasks involving computer animation, and (Cleveland & McGill, 1986; Petre & Green, 1993) for studies of comprehension tasks involving graphics.

[5]Notably, while programs that are to be visualized using computer-based systems must be implemented in a programming language, programs that are to be visualized using pen, paper, or other "low tech" artifacts may not be actually implemented at all. In the latter case, one may rely on a pseudocode description of the program, or on one's conceptual model of how the program works (see Chaabouni, 1996, ch. 5).

Brown's (1988a) *interesting events* paradigm, this task involves annotating the program source code at the points of interesting events. Under Roman, Cox, Wilcox, and Plun's (1992) declarative scheme, in contrast, this task involves defining a set of rules that map program data and events to the proof mapping space. In either case, the essence of the task is (a) to identify the abstractions in terms of which the visualization will depict the program to be visualized, and (b) to summon the program data necessary to depict the program in terms of those abstractions.

2.3 Visualization view writing

Given events and data that are of interest to the visualization, one completes the mapping process by engaging in *visualization view writing*. In this task, one writes general routines that produce *visualization views*—that is, graphical representations of the data and events of interest. In SV systems that offer non-customizable views of programs (e.g., Brayshaw & Eisenstadt, 1991), visualization view writing is not performed by system users. In other systems, visualization view writing may be performed, but it may require low-level, device-specific graphics programming, discouraging all but the most experienced programmers from performing it (e.g., Brown, 1988*a*). On the other end of the extreme, some systems require users to engage in visualization view writing, since no predefined views are offered (e.g., Mukherjea & Stasko, 1994).

2.4 Input data selection

In order to generate a visualization, one needs to execute the program on a set of input data. At the time of algorithm execution, one engages in *input data selection* by choosing the set of input data to use. Some systems allow the end-user to select the input data as part of a visualization exploration process (e.g., Brown, 1988*a*), while, in others, the input data are selected well in advance of visualization exploration (e.g., Baecker, 1981).[6]

2.5 Visualization exploration

The visualization exploration task is the crux of the visualization process, for it is the task in which humans presumably reap the benefits of SV. In this task, the *visualization explorer* manipulates the visualization's user interface to explore the visualization. Visualization exploration fundamentally involves the *viewing* of a visualization. In addition, a SV system's user interface may support additional kinds of exploration. In the seminal work on SV user interfaces, Brown (1988*a*) suggests four kinds of view exploration to be supported:

- *visualization execution control*, by which visualization explorers start, stop, and step through the visualization, and adjust the visualization speed;
- *view selection and arrangement*, by which visualization explorers select views of the software to watch from a list of available ones, and arrange those views on the screen; and
- *view zooming and panning*, by which visualization explorers select the scale factor of a view, as well as the region of the view to watch.

Subsequent SV research has expanded that repertoire to include so-called *semantics-based interactions* (Hundhausen, 1995), such as altering the data and execution path of programs (see, e.g., Brown, 1991; Gu *et al.*, 1994) and semantic zooming (see, e.g., Stasko & Mutukumarasamy, 1996).

It is important to note that the name of this task, *visualization exploration*, is slightly misleading, for it implies that the visualization itself, rather than the explorer's own *task goal*, is the focus of the task. For example, consider the debugging task examined by Price (1990). In his study, participants

[6]cf. Chaabouni's (1996) description of how humans build visualizations using art supplies. Her observations indicate that Event and Data Selection, Visualization View Writing, and Input Data Selection occur *dialectically*— in no particular order. In fact, the input data are often selected *before* the visualization is actually designed.

enlisted SV not for its own sake, but rather to fulfill the goal of the task: namely, *to find a bug in a program*. Indeed, as studies of the use of visual representations in other domains have shown (see, e.g., Petre, 1995), visualizations are not paintings; rather, visualization explorers always explore for some *purpose*, which is rooted in the task in which the visualization explorer is enlisting SV, not in the visualization itself.[7]

2.6 Visualization narration

In many scenarios [e.g., those suggested by (Baecker, 1981) and (Brown, 1988*a*, Appendix A)], a visualization is not merely interacted with; rather, it is used as a resource in a presentation or explanation. In such cases, *visualization narration* becomes a task in itself.[8] While it might be seen as just a special form of visualization exploration, visualization narration appears to be so widely used, and so important to effectiveness in many scenarios, that it merits recognition as a separate task. For example, imagine what it would be like to view the film *Sorting Out Sorting* (Baecker, 1981) with the volume turned down. Without its adroit use of narration, both to complement and to illuminate the concomitant algorithm animations, it seems unlikely that the film would have had such widespread appeal.

PROGRAMMING	DATA/EVENT SELECTION	VIEW WRITING	INPUT DATA SELECTION	VIEW EXPLORATION	NARRATION
What program shall I visualize, and how shall I write it?	What aspects of the program shall I visualize?	How should I represent those aspects?	On what input data shall I visualize the program?	How shall I explore the visualization for the problem at hand?	How shall I narrate what's going on in the visualization?

Table 2. Summary of the tasks of SV, in terms of key questions that performers of each task must ask

2.7 Tying the tasks of SV together

Table 2 provides a concise summary of the six tasks of SV. The important question arises, Aside from the obvious connections implied by the formal model of SV (Figure 2), how are these six tasks related? To answer that question, I define the notion of a scenario of SV use:

> **Scenario of SV use**: An instance in which a group of people, each of whom takes on one or more of the tasks of SV, make use of a specific SV artifact to explore a target program for some overriding purpose.

It is important to underscore three observations with respect to this definition. First, notice that any time one chooses to use an SV system to explore a target program (i.e., engage in View Exploration[9]), the first three tasks of SV must have already been completed. It may be the case that those tasks were completed well in advance of the actual View Exploration session [e.g., in the case of scenarios of SV use suggested by Baecker's (1981) *Sorting Out Sorting* film], but the fact remains that someone had to perform them. Second, observe that, within a given scenario of SV use, the tasks of Input Data Selection and Narration may not occur at all. Indeed, some programs do not have input data, and not all scenarios of SV use have a need for narration. Finally, the SV systems

[7]Of course, visualization exploration might lead to the formation of new, unanticipated goals that relate only indirectly to the task at hand. For example, if the visualization's user interface gets in the way, the visualization explorer may have to divert attention from her original goal in order to figure out the interface. I should note that activity theory (see Nardi, 1996) accounts for these two levels of of computer-mediated activity as the *handling aspects* and the *subject/object-directed aspects* (Bødker, 1996). However, a more thorough treatment of the issues involved in these multiple levels of activity is beyond the scope of this meta-study,

[8]See (Chaabouni, 1996, ch. 5) for a detailed analysis of SV narration and its relation to the objects of a SV.

[9]In order to emphasize that they have been precisely defined, I capitalize the names of the six tasks of SV throughout the article.

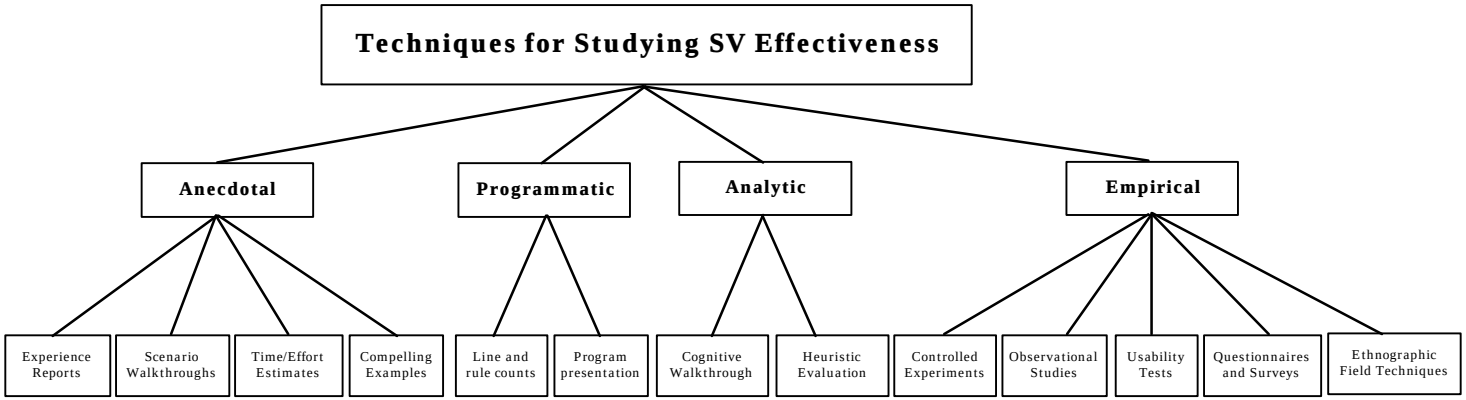

Figure 3. A taxonomy of approaches to studying SV effectiveness

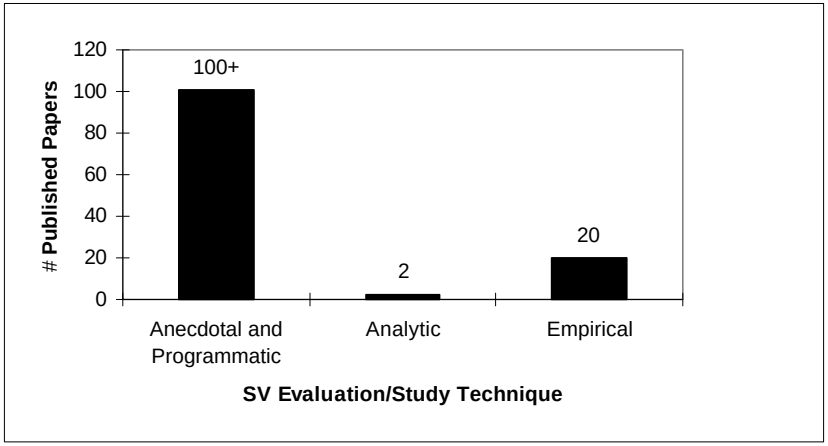

Figure 4. Summary of published papers on SV effectiveness by technique. Published papers that employ more than one technique are included multiple times—once in each appropriate category.

literature (see, e.g., Brown, 1988*a*) suggests both that the same person seldom engages single-handedly in all of the tasks of SV within a given scenario, and that each individual SV task is seldom an individual endeavor. Rather, people often engage *collaboratively* in SV tasks, both at the level of single SV tasks, and at the level of the entire SV process depicted in Figure 2.

We see, then, that *who* actually performs the tasks of SV, *when* they are performed, and for *what overriding purpose* are the three fundamental questions to be answered with respect to a scenario of SV use. As we shall see in the review of studies of SV tasks that follows, most of the extant literature has focused on single individuals performing isolated SV tasks, without revealing the details of the scenarios of SV use within which those tasks are performed.

3. Data and observations: A review of research into SV effectiveness

The task-based definition of SV outlined in the previous section circumscribes a body of research with a common interest in studying, evaluating, and ultimately improving the effectiveness of SV technology within human tasks. I now turn to the data upon which this meta-study is based: a comprehensive review of that body of research.[10] The purpose of the review is to set the stage for the analyses of Section 4. Thus, I have aimed keep the it as neutral as possible, providing just enough detail to serve the purposes of my ensuing analyses.[11]

All of the literature considered in this section makes use of one or more of the techniques presented in the taxonomy of Figure 3. Figure 4 graphs the number of published papers that employ each of the taxonomy's four top-level techniques. As can be seen from that figure, papers that make use of the *anecdotal* or *programmatic* techniques outnumber the others by at least five to one.[12] Each sub- section below follows one of the four top-level branches downward. For each leaf-level technique along a given branch, I briefly introduce the technique, and I provide a synopsis of the published research into SV effectiveness that has employed that technique. The section concludes with a brief summary of the review.

[10]An appendix at the end of this article consolidates all of the meta-study's principal data for easy access. Readers interested in further scrutinizing the numerical counts that are graphed and discussed in Sections 3 and 4 can consult the appendix, which indicates the precise manner in which each of the SV effectiveness studies is classified.

[11]Nonetheless, I realize that any review is colored by the perspective and biases of its author. Rather than deny that bias, I believe that it is important to make it explicit. My choice to view SV as a set of tasks, rather than as a technology, plainly indicates my belief that effectiveness cannot be considered on a system-by-system basis, as suggested by Price, Baecker, and Small's (1993) taxonomy. Rather, I hold that SV effectiveness must be grounded in a specific SV task, as defined in the previous section. As a consequence, the anecdotal and programmatic techniques described below can, in my view, be dismissed rapidly as viable means of studying SV effectiveness.

[12]Because of the formidable difficulty of providing precise counts of papers that have employed the *anecdotal* and *programmatic* techniques (the number of such papers is prodigious), and because of this study's ultimate focus on *empirical* techniques, Figure 4 groups together the papers containing anecdotal and programmatic evidence of effectiveness. To arrive at the figure for the number of papers containing anecdotal or programmatic evidence of effectiveness, I have made a simplifying, but reasonable, assumption: namely, that all SV systems reported in the literature have minimally been evaluated either anecdotally or programmatically. This implies that the number of papers containing anecdotal or programmatic evidence of effectiveness necessarily corresponds to the number of published SV systems papers. In their 1993 taxonomy, Price, Baecker, and Small (1993) estimate the number of prototype SV systems to be over one-hundred. It follows, then, that at the time of this writing, the papers containing anecdotal or programmatic evidence of effectiveness number at least one hundred. In actuality, the number of such papers is probably much larger than one hundred, since it is often the case that several papers are written on a single SV system.

3.1 Anecdotal techniques

Authors of SV system papers typically include a section in which they aim to convince readers that their SV system not only is functional, but also has been successfully applied to an interesting problem. *Anecdotal* evaluation techniques aim to appeal to reader intuition by presenting instances and examples of system use, as recounted by the system's authors.

I distinguish four kinds of anecdotal evaluation. Perhaps the most valuable form of anecdotal evaluation, the *experience report* (see, e.g., Stasko, 1996) recounts authors' personal experience with their system, in an attempt to provide readers with advice on how it can best be used. *Scenario walkthroughs* (see, e.g., Chapter 3 of Brown, 1988a) describe the use of a system under a hypothetical, but highly plausible, set of circumstances. *Time and effort estimates* indicate an author's best guess of how long a user can expect to spend completing a task with the SV system, such as programming a new view (see, e.g., Brown and Sedgewick, 1985). Finally, in order to highlight actual instances in which their system clearly succeeded in assisting its users, authors of system papers often present c*ompelling examples* of their system in use (see, e.g., Kimelman, Rosenburg, & Roth, 1994).

Nearly all of the technical literature on prototype SV systems contains some sort of anecdotal evaluation. As of 1993, over one hundred such systems had been built (Price, Baecker, & Small, 1993). Under the highly conservative assumption that just one paper was published on each prototype system, the number of anecdotal evaluations of SV found in the literature thus numbers in the hundreds. Consequently, an exhaustive review of anecdotal evaluations would need to consider a gargantuan body of literature, and is well beyond the scope of this review. For the purposes of this article, it suffices to point out that the overwhelming majority of anecdotal evaluation places SV systems in an unabashedly positive light; indeed, to do otherwise would be to undermine anecdotal evaluation's largely rhetorical purpose.

3.2 Programmatic techniques

Unlike anecdotal evaluation, which relies on human memory and rhetorical skill for persuasive power, *programmatic techniques* rest on publicly available evidence: the actual programs used to produce visualizations within a given SV system. *Line and rule counts* simply sum the number of lines of code, or number of rules, that are sufficient to specify a visualization. The lower that number seems with respect to the complexity and sophistication of the visualization, the more favorable the evaluation. For example, Cox and Roman (1994) make extensive use of rule counts in their evaluation of the Pavane system (Roman *et al.*, 1993), presenting tables that list ranges of sufficient rule counts for several algorithms, including quick sort (2 – 12 rules), a shortest paths algorithm (2 – 11 rules), and a matrix transposition algorithm (5 – 10 rules).

In a similar vein, the *program presentation* simply makes available for public inspection the actual programs used to produce visualizations. The more readable the program appears with respect to the visualization it produces, and the closer the apparent correspondence between the program and the visualization it produces, the more positive the evaluation. Stasko (1989), for example, includes an entire appendix of algorithm animation source code, which he uses to illustrate the extent to which his programming framework succeeds in easing the task of algorithm animation programming.

Although far less popular than anecdotal techniques, programmatic techniques—especially program presentation—do appear on occasion in the SV systems literature. Programmatic techniques tend to scrutinize programs that generate visualizations of textbook algorithms (but see Cox & Roman, 1994). Like anecdotal techniques, programmatic techniques serve a largely rhetorical purpose; hence, the results of programmatic evaluations are overwhelmingly positive, at least from the perspective of their authors.

3.3 Analytic techniques

Analytic evaluation techniques aim to provide a principled assessment of an interactive system's effectiveness, while avoiding the overhead of extensive empirical data collection. Effectiveness, in the case of analytic evaluation, boils down to *usability*—the fewer usability problems identified, the more effective the system. Thus, analytic evaluation techniques focus on identifying the usability problems that system users are likely to encounter.

In *heuristic evaluation*, a user interface expert sits down with an interactive system and attempts to evaluate it using a small set of design principles. The success of analytic techniques appears to turn heavily on the expertise of the evaluators, on the evaluators' familiarity with the domain of the interactive system under evaluation, and on the number of evaluators involved (three to five are preferable to one) (Nielson, 1992).

For the only heuristic evaluation of an SV system in the literature, Lavery and Cockton (1995) consolidated seven relevant design principles from three sources: (1) the general information presentation literature (e.g., Tufte, 1983) (three principles); (2) the general visualization literature (three principles); and (3) the SV literature (one principle). In their heuristic evaluation of two SV prototypes designed to aid programmers using a persistent software engineering workbench, they find that the design principles were too general, providing little helpful guidance in their evaluation.

In a *cognitive walkthrough* (Polson, Lewis, Rieman, & Wharton, 1992), an individual evaluator, or group of evaluators, attempt to simulate the activities of a user as she explores an unfamiliar user interface. Beginning with a specific set of tasks to be performed with the system under evaluation, the evaluators look for user interface actions that appear promising in accomplishing those tasks. If they can find suitable user interface actions, they select them and evaluate the system's feedback. At each point along the way, they consider three questions: (a) whether users will be able to come up with the a set of correct actions to complete the tasks; (b) whether users will be able to link descriptions of the correct actions to the actions they have in mind; and (c) whether users will interpret the system's feedback correctly.

Just two applications of the cognitive walkthrough technique appear in the SV literature. In the wake of disappointing experimental results (see Section 3.4.1 below), Stasko, Badre, and Lewis (1993) use a cognitive walkthrough to determine that their participants could not have been expected to perform well on the experiment's post-test, given the XTango animation of the pairing heap data structure with which they were presented. Frustrated with the failure of their heuristic evaluation (see above) to yield useful design information, Lavery and Cockton (1995) use an informal cognitive walkthrough to analyze a set of specific tasks to be performed with the same two SV prototypes. In contrast to their heuristic evaluation, their cognitive walkthrough enables them to identify several potential usability problems, which account for forty percent of the problems identified in a follow-up usability test (see Section 3.4.2 for more on usability tests).

3.4 Empirical techniques

In contrast to the three techniques just discussed, empirical evaluation techniques are predicated upon the collection of actual data on humans involved in tasks with SV systems. An analysis process, whose details are made available for public scrutiny, attempts to transform the data into a set of statements that respond to the research questions posed by the evaluation. Thus, the audience of an empirical evaluation is able both to trace the evaluation's conclusions back to original data, and to scrutinize the process by which the conclusions were arrived at.

Figure 5 provides a breakdown of the published empirical studies of SV by empirical technique. Since relatively few total empirical studies have been published, the meta-study's sample of 29 SV empirical studies either matches, or closely rivals, the entire population. It is common for studies to

employ multiple techniques; hence, several studies are counted multiple times—once in the total of each appropriate category.

As Figure 5 indicates, the number of published experimental studies using a given technique ranges from eleven to two, with *controlled experiments* and *observational studies* tied for the lead. This subsection takes up the five empirical techniques in decreasing order of their frequency of use in the literature.

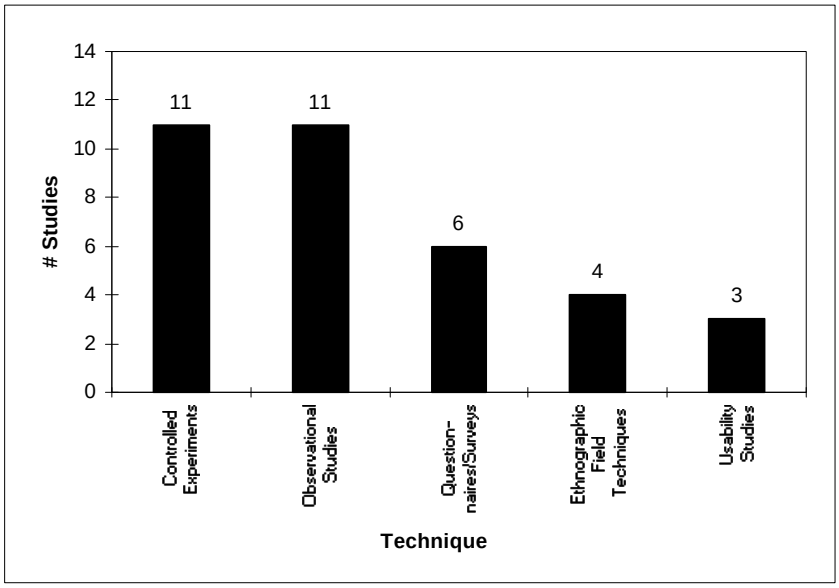

Figure 5. Summary of published empirical studies of SV by study technique. Note that *study* does not equate to *paper*; a single piece of published research may report on more than one study. Studies that employ more than one technique are included multiple times—once in each appropriate category.

3.4.1 Controlled experiments

One of the most popular empirical techniques for evaluating SV, *controlled experiments*[13] aim to assert a causal relationship between factors (i.e., independent variables) and measures (i.e. dependent variables). While there are many variants on controlled experiments (see Gilmore, 1990 for a review), all of the published SV experiments have been *between-subjects experimental comparisons*. In such comparisons, participants are first screened in an attempt to ensure that they have comparable backgrounds, experience, and abilities. Next, experiment participants are randomly assigned to one of two groups, each of which is exposed to a different combination of factors that the experimenters believe to have significant effects. Third, participant performance is measured. In order for such measurements to be made, experimenters must *operationalize* the dependent variables of interest—that is, they must express them in terms of observable and measurable phenomena. Finally, in order to determine whether any measured differences are greater than those to be expected by random chance, experimenters statistically compare the measures of the alternative groups. If statistically significant differences are detected, experimenters conclude that the factors significantly affect the measures.

Table 3 provides a synopsis of all eleven controlled experiments that have considered SV effectiveness. For each experiment, the factors (independent variables) appear in column 1; the measures

[13]This discussion of controlled experiments by no means addresses all of the issues involved; it does, however, provide sufficient background on the kinds of controlled experiements of SV that have been conducted thus far.

(dependent variables) appear in column 3; the programs visualized by the experiment participants, together with the SV artifact they used, appear in column 4; and a summary of the experiment's key results appears in column 5.

As Table 3 indicates, nine of the experimental comparisons attempt to determine whether various factors affect *learning*. In all of these experiments, learning is operationalized in terms of a post-test, which participants take upon completing their learning session. The post-test is typically designed to test participants' procedural ("how") and declarative ("what") knowledge of the algorithm under study in the experiment. The more correct answers provided by a participant, the better the participant is assumed to have learned the algorithm.

Similar in spirit to the experimental comparisons of learning, two other experiments consider whether SV tracers facilitate better problem solving. In these experiments, problem-solving efficacy is operationalized and measured in various ways. In one experiment (Price, 1990), problem-solving efficacy is defined both in terms of whether the bug in the program was found, and in terms of how long participants needed to find the bug. In the other experiment (Mulholland, In press), problem-solving efficacy is defined in terms of the number of problems solved (out of a possible four).[14]

In the eleven experiments, participants use a total of four different SV systems (ParaDocs, XTango, Polka, and TPM) to visualize algorithms and data structures one would typically find in a textbook. The notable exception is Price's (1990) experiment, in which participants debug a 7,500 line program.

The eleven experiments consider a total of twelve factors, which fall into five general categories:

- *medium*—algorithm animation, visual debugger, text, and various combinations of media;
- *order*—order of algorithm presentation, order of medium;
- *representation*—e.g., data labeling, data representation, data set size;
- *learner activity*—self-design of data sets, prediction drills; and
- *individual factors*—learning ability, spatial ability.

Of the 12 factors considered, five of them, or 42%, are shown to affect learning or debugging efficacy significantly. Two of the factors, both of which fall within the *learner activity* category, have gathered more experimental support than any other factors. In two separate experiments, Lawrence (1993, ch. 6 and 9)[15] finds that students who explore algorithm animations driven by self-constructed input data sets score significantly higher than students who either watch the same animations driven by data supplied by the experimenter, or who have no access to such animations. In two other experiments that manipulate learner activity variables, Byrne, Catrambone, and Stasko (1996) find that participants who are exposed to algorithm animation, combined with structured prediction drills, perform better on a post-test than students who are exposed to neither; however, the individual effects of the two factors cannot be disentangled statistically.

Lawrence (1993, ch. 7) shows that two representation factors—labeling of an algorithm's conceptual steps, and additional color labeling of algorithmic actions—significantly affect post-test performance. Students who saw animations of Kruskal's minimum spanning tree algorithm in which the algorithm's conceptual steps were textually labeled performed significantly higher than students

[14]The most significant portion of Mulholland's (In press) work is a post-hoc analysis of videotaped participant interaction; this analysis leads to the discovery of three additional significant differences. Since Mulholland operationalizes post hoc the categories between which significant differences are found, however, I have elected to include these results in my review of observational studies (Section 3.4.2).

[15]Note that (Lawrence, 1993, ch. 9) is also published as (Lawrence, Badre, & Stasko, 1994).

STUDY	INDEPENDENT VARIABLES	DEPENDENT VARIABLES	PROGRAM(S)/ SV ARTIFACT(S)	RESULTS
(Price, 1990, Experiment)	• *Debugging medium* (debugging with animated view vs. debugging without animated view)	• Debugging time • Whether bug found	7,500 line Operating System Simulator/ParaDocs (Price, 1990)	• No significant differences were found: An equal number of participants (5 of 9) in animation and control groups found bug, and the mean time to find bug was identical (25 min.) for animation and control groups
(Stasko, Badre, & Lewis, 1993)	• *Learning medium* (text only vs. text-and-animation)	• Post-test scores	Pairing heap ADT/ XTango (Stasko, 1992)	• Non-significant trend favoring the text-and-animation group ($t=1.111$, $df=18$, $p<0.13$) • Cognitive walkthrough indicated that not all information necessary to do well on test was available to either group
(Lawrence, 1993, ch. 4.4, 5, 6, 7, 8.2, 9)	• *Data set size* (9, 25, or 41) • *Data representation style* (horizontal sticks, vertical sticks, and dots) • *Order of algorithm presentation* (Quick sort first vs. Selection sort first) • *Spatial ability* (according to a paper-folding test) • *Verbal ability* (according to a vocabulary test) • *Learner control of data* (viewing self-designed data sets vs. viewing data sets designed by someone else) • *Algorithm step labeling* (color vs. monochrome highlighting of algorithm's operations, as well as text vs. no labels of algorithm's conceptual steps) • *Order of medium* (text-first vs. animation-first) • *Combination of learning environments* (lecture-only vs. lecture-and-lab)	• Paper-folding test • Vocabulary test • Post-test scores • Time to complete post-test	Quick sort, selection sort, radix sort, and Kruskal's MST/ XTango (Stasko, 1992) and POLKA (Stasko & Kraemer, 1993)	• Participants who viewed animations running on their own data sets scored significantly higher than (a) participants who viewed animations running on data sets designed by someone else (ch. 6), and (b) participants who had no access to animation sessions (ch. 9); in the experiment of ch. 9, the significant difference was detected for the conceptual "free response" questions • Participants who viewed animations containing no color labeling scored significantly higher than participants who viewed animations with color labeling (ch. 7) • Participants viewed animations containing conceptual step labels scored significantly higher than participants who viewed animations containing no conceptual step labels (ch. 7) • No other significant differences were detected. Notably, the visual representation of animations did not seem to matter.
(Byrne, Catrambone, & Stasko, 1996, §2 and §3)	• *Learning medium* (animation vs. no-animation) • *Prediction* (as manifested in a prediction drill in which participants are asked to predict the next algorithm step, given the current step)	• Post-test scores • Prediction errors	Depth-first search and binomial heaps/POLKA (Stasko & Kraemer, 1993)	• Participants who viewed the animation and/or made predictions performed significantly better than participants who did neither. This result was stronger for novices learning depth-first search (§2) than for upper-level computer science students learning the binomial heap (§3). In the experiment of §2, the significant difference was found for the "hard" questions. In the experiment of §3, the significant difference was detected for the "procedural" questions.
(Mulholland, in press)	*Prolog tracing environment* (three textual tracers, TPM)	Number of problems solved (5 min. max per problem)	A short Prolog program/ TPM (Brayshaw & Eisenstadt, 1991)	• Participants who used the graphical tracer (TPM) solved significantly fewer problems than participants who used the textual tracers (Spy, PTP, TTT)

Table 3. Summary of controlled experiments that consider the effectiveness

who saw the same animations without such labels. In addition, the experiment shows that the addition of color highlighting[16] actually leads to decreased performance; those who saw animations with such labels performed significantly worse than those who saw animations without such color highlighting.

Finally, Mulholland (In press) actually obtained evidence against the use of SV for Prolog tracing tasks. He finds that participants who used any of three textual tracers solved significantly more problems than participants who used TPM, a graphical tracer.

Figure 6 places the results of the controlled experiments into perspective, indicating the percentage of experiments that have obtained each kind of the four kinds of results just reported. As the figure indicates, over half of the controlled experiments (55%) either failed to produce a statistically-significant result, or produced a statistically-significant result in the wrong direction—that is, a result that favored some other medium over SV. Of the remaining five, only three could assert a positive effect that was not entangled with another factor.

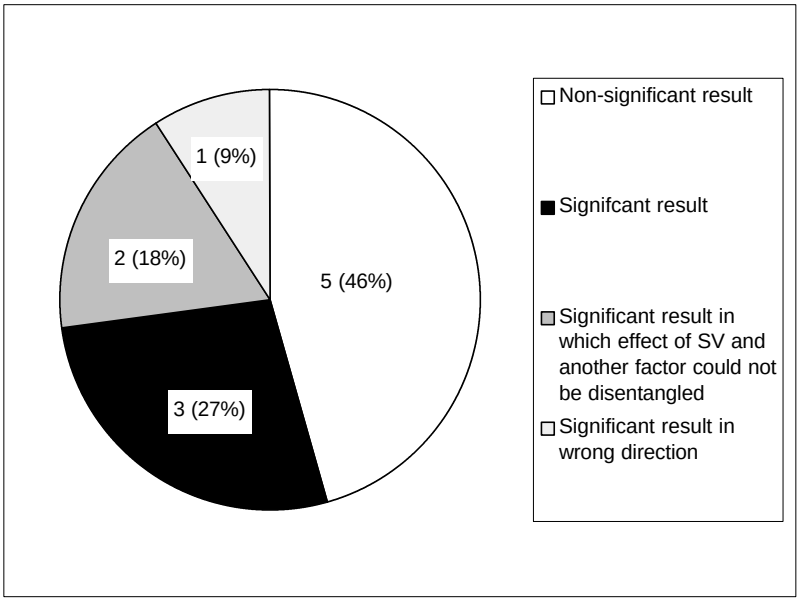

Figure 6. Summary of the kinds of quantitative results yielded by the SV controlled experiments

3.4.2 Observational studies

Less rigorous than controlled experiments, *observational studies* investigate some activity of interest in an exploratory, qualitative fashion.[17] Customarily, observational studies have one of two goals. They may explore phenomena about which little is known, with the goal of generating research questions and formulating hypotheses that can guide the design of more rigorous controlled experiments. Alternatively, they may be used to explore research questions that rigorous experiments are ill-suited to address—for example, questions of *how* a process unfolds, or questions concerning the nature of social interaction, which is inherently difficult to operationalize.

Table 4 summarizes the eleven observational studies of SV tasks. As the table indicates, all of the observational studies seem to be interested in questions of *how* humans make use of SV, and of how SV can be improved. In fact, the foci of all of the studies appear to be subsumed by four general research questions, as indicated in Table 5.

[16]The highlighting is used to indicate which edges have not yet been considered (blue), which edges are currently being considered (red), and which edges are already in the minimum spanning tree (black).

[17]See (Gilmore, 1990) for a more comprehensive treatment of observational techiques.

STUDY	RESEARCH QUESTION(S)	PROGRAM(S)/ SV ARTIFACT(S)	DATA COLLECTED	RESULTS
(Price, 1990, Experiment)	• How much progress did those who did not find the progress make? • What strategies do people employ to track down bugs using SV and conventional debugging media? Do those strategies depend upon debugging medium?	7,500 line Operating System Simulator/ ParaDocs (Price, 1990)	Videotape of debugging session, exit interviews	• Three in the animation group, and one in the control group, were "close" to finding the bug • Debugging strategy (skilled vs. unskilled) appeared to be influence debugging success more than debugging medium.
(Badre, Baranek, Morris, & Stasko, 1991, §2.2)	• How can we evaluate algorithm understanding? • What problems might arise in such evaluations? • What constitutes a good algorithm animation learning environment?	XTango/Shellsort	Observations of student activity, questionnaire, informal exit interviews	• Students' reactions to animations were overwhelmingly positive • May be difficult to disentangle individual factors that influence learning, including academic background, spatial abilities, and prior experience; future experiments will need to control for these • The effectiveness of animations might be enhanced by adding labels and textual explanations
(Goldenson & Wang, 1991),	• How do students make use of Pascal Genie's environment features, including the following SV features: edit-time design view, edit-time outline view, run-time stack view, and run-time data visualization views? • What factors influence students' use of environment features?	Student assignments/ Pascal Genie (Chandhok *et al.*, 1991)	Software event logs, filtered to the level of Pascal Genie's environmental semantics	• Teachers' knowledge of and enthusiasm for Pascal Genie's SV features heavily influenced their students' likelihood of using them. • Students used the run-time SV features of the environment to help them find program errors; however, they did not seem to use them as often as their errors would have justified
(Ford, 1993)	• How do learners' conceptions and misconceptions of procedural and object-oriented programming constructs manifest themselves in visualizations?	Imperative programming constructs and concepts/ Goofy	Observations of student-built Goofy animations, interviews	• Students expressed imperative and object-oriented programming constructs using a rich visual vocabulary • Of the visualization techniques described in Cox and Roman's (1992) taxonomy of representations, students made use of all but *analytical* representations • The process of constructing visualizations appeared to help students learn the language
(Lawrence, 1993, ch. 8.3)	• Exploratory between-subjects Design: How does media combination (text-only, animation-only, or text-and-animation) affect learning?	Selection sort, radix sort, and quick sort/XTango	Observations of student activity, number of correct algorithm rules derived	• Text-only group scored lowest for selection sort, although all three groups scored similarly for radix sort; this implies that algorithm may influence appropriateness of medium
(Wilson, Katz, Ingargiola, Aiken, & Hoskin, 1995)	• What role do two alternative algorithm animation displays play in learning about search algorithms?	Breadth-first and A* algorithms/ FLAIR (Aiken *et al.*, 1992)	Videotape of activity of a pair of participants	• Because it made readily available to them the information they needed to complete their task, participants found one display more useful than the other

Table 4. Summary of the studies that make use of observational techniques

STUDY	RESEARCH QUESTION(S)	PROGRAM(S)/ SV SYSTEM(S)	DATA COLLECTED	RESULTS
(Douglas, Hundhausen, & McKeown, 1995, 1996)	• How do experts use visualization to explain bubble sort? • How can SV languages be grounded in empirical data? • Does the usability of SV systems depend on their ability to offer a visualization language that accords with human conceptualizations of the computations to be visualized with the systems?	Bubble sort/ Art supplies and Lens (Mukherjea & Stasko, 1994)	Videotape of participants' visualization storyboards, and of Lens sessions	• At a lexical level, expert visualizations of bubble sort vary widely; they can be unified, however, at a semantic level • Incongruities between participants' semantic-level languages and that offered by Lens led to usability problems
(Chaabouni, 1996)	• How do experts use visualization to explain sorting algorithms? • What conceptual primitives do they use? • How can those primitives be transformed into a visualization language, and what does such a language look like?	Quick sort, heap sort, and insertion sort/ Art supplies	Videotape of pairs of participants' visualization storyboards, exit questionnaire	• Participants skillfully orchestrated their use of speech, gesture, and environment objects to explain sorting algorithms • A semantic-level ADT language consisting of six ADTs and 58 ADT operations is sufficient to express all nine pairs of participants' visualizations; however, it may not be sufficiently expressive for a wider range of computations than those studied
(Kehoe & Stasko, 1996	• What information do students obtain from algorithm animations, text, and static pictures? • Do alternative learning media provide students with the information they are seeking? • When in the learning process do students enlist algorithm animation?	Binomial heap implementation of pairing heap/ A web-based tutorial with POLKA stills	Videotape of participant activity (participants asked to think aloud)	• Students use animation to learn about procedural steps of an algorithm • User interface, rather than the animation itself, can be a barrier to obtaining desired information • Animations were used both to gain an overall understanding of algorithm's procedural behavior, and to refine knowledge of a specific operation
(Reimer, 1996)	• Can a principled, task-centered design approach result in more effective algorithm animations? • Can principle of *semantic redundancy* assist in the design of effective visualizations?	Graham's scan convex hull/ HyperCard	Observation of participant activity (participants asked to think aloud), post-test scores, exit questionnaire	• Participants enjoyed using the animations to learn about the algorithm • Encoding salient information redundantly (both textually and graphically) in an animation appears to improve understanding
(Mulholland, In press)	• How do Prolog tracing environments affect novice debugging strategies?	A short Prolog program/ TPM (Bayshaw & Eisenstadt, 1991)	Videotape of pairs of participants' task sessions, post hoc behavioral coding	• TPM participants made significantly more comments on trace notation and navigation, and significantly fewer comments on control and data flow, than did participants using the textual tracers • TPM participants reviewed previous executions steps and data flow significantly fewer times than participants using the PTP tracer. • TPM participants experienced significantly more trace misunderstandings than did participants using the PTP tracer

Table 4 (cont.). Summary of observational studies of SV

GENERAL RESEARCH QUESTION	# OBSERVATIONAL STUDIES THAT CONSIDER IT	
How do humans conceptualize algorithms?	3	(Ford, 1993; Douglas *et al.* 1996; Chaabouni, 1996)
How frequently is SV used, and what role does it play, in various tasks?	5	(Goldenson & Wang, 1991; Price, 1990; Kehoe & Stasko, 1996; Wilson *et al.*, 1995; Mulholland, In press)
How can empirical data be used to improve the design of SV systems and languages?	3	(Douglas *et al.*, 1995; Chabouni, 1996; Reimer, 1996)
How can algorithm understanding be evaluated?	1	(Badre *et al.*, 1991)

Table 5. Four general research questions considered by the observational studies of SV, along with the number of studies (and the studies themselves) that have considered them

With the exception of Goldenson and Wang's (1991) analysis of Genie user profile data and Price's (1990) study of debugging, all of the observational studies focus on compact, textbook algorithms or programming constructs. As seems to be the case for the controlled experiments, such a focus stems from researchers' common interest in the use of SV for education. Six SV systems were used in the observational studies. In addition, three studies chose to abandon SV technology altogether, relying instead on storyboards of animations developed with pen, paper, and art supplies.

Data collection methods vary widely from study to study. In all but one case, researchers gave participants explicit tasks to perform, and the participants performed those tasks within a single task session. In about half of the studies, researchers chose to capture participant activity on videotape; in the other half, researchers opted simply to observe and take notes. In stark contrast to the other eight studies, the Goldenson and Wang (1991) study is notable for its use of special software to log and filter user events for post-hoc statistical analysis.

While the majority of the SV observational studies rely on informal analysis techniques to draw conclusions, a few studies employ more formal techniques. In particular, Douglas *et al.* (1995, 1996) make use of *conversation analysis* (Douglas, 1995); Price (1990), Kehoe and Stasko (1996), and Mulholland (1996) employ *protocol analysis* (Ericsson & Simon, 1984); and Chaabouni (1996) enlists *interaction analysis* (Jordan & Henderson, 1994). Each of these techniques advocates a slightly different set of analytical foci, and aims for results of a slightly different flavor; see (Sanderson & Fischer, 1994) for a taste of the tradeoffs involved.

As the unified research questions in Table 5 suggest, the observational studies share a common interest in exploring how and why humans make use of SV in various SV tasks. Their ultimate goal, as explicated by the third question in the table, is to apply observations to the (effective) design and use of SV technology. Table 6 summarizes the results of the observational studies on a SV task-by-SV task basis. For each task, the table synthesizes the observational studies' conclusions with respect to how and why SV is used. Any implications the findings have for SV design and use are also listed.

3.4.3 Questionnaires and surveys

Often used as a complementary source of data in empirical studies, *questionnaires* and *surveys* elicit written responses to a set of questions in which the researcher is interested.[18] Most frequently, surveys and questionnaires request subjective data on their respondents' preferences, opinions, and advice.

[18]Some might contest my choice not to include *interviews* in this category. Indeed, like questionnaires and surveys, interviews are just another way of eliciting responses to a set of questions in which one is interested. On the surface, the only difference appears to lie in the medium; whereas questionnaires and surveys elicit *written* responses, interviews elicit *verbal* responses. However, because of the interview's prominence and heritage as an ethnographic field technique (see Spradley, 1979), I decided in the end to include it in that category, with the recognition that this classification may not be agreeable to everyone.

As indicated in Sections 3.4.1 and 3.4.2, several controlled experiments and observational studies make use of questionnaires and surveys to collect *complementary* data. Such data are used in two general capacities. First, in the controlled experiments, screening questionnaire data are used in

SV TASK	HOW IS SV USED?	WHY IS SV USED?	IMPLICATIONS FOR SV DESIGN AND USE
Programming (Goldenson & Wang, 1991)	• People do not use SV views of programs as frequently as would be predicted by the number of bugs in their programs	• People use SV views to find and correct bugs • Teacher knowledge and encouragement influences whether students will use SV views	• Teachers interested in using programming environments containing SV features need to be enthusiastic about those environments themselves
Data/event selection and view writing (Douglas *et al.*, 1995, 1996; Chaabouni, 1993; Ford, 1993)	• Given similar visualization programming tasks, peoples representations will vary markedly at the lexical level; however, semantic-level differences will be minimal • Perceived audience drives people's choices of what to depict in a SV	—Not applicable—	• SV languages should provide mechanisms for supporting the common semantics of the algorithmic domain, while providing programmers with lexical flexibility
SV exploration (Price, 1990; Badre et al., 1991, §2.2; Wilson *et al.*, 1995; Kehoe & Stasko, 1996; Mulholland, In press)	• People use SV in concert with other representations • People use SV to help them trace through algorithms • People use an SV tracer to obtain an overview of program execution, but may have difficulty using it to obtain fine-grained information	• SV can make readily available the information needed to perform a task • SV appears to help people come to grips with the details of an algorithm's procedural steps.	• Environments should attempt to make explicit connections among representations • SV is useful in explaining procedural behavior of algorithms
SV narration (Douglas *et al.*, 1996; Chaabouni, 1996)	• People skillfully coordinate gestures, speech, and SV in their explanations of algorithms • Narration amounts to a play-by-play of a SV, which simulates the algorithm running on a sample data set	• SV gives the important concepts of an algorithm a publicly-available physical form, allowing those concepts to be discussed and illuminated	• The importance of speech and gesture in the explanation of SV should not be overlooked

Table 6. Summary of observational study findings by SV task.

STUDY	ROLE	RESPONDENTS	PURPOSE	FINDINGS
(Badre, Baranek, Moris, & Stasko, 1991, §2.1)	Primary	11 CS professors at two universities	To elicit data on how professors teach and conceptualize algorithms?	• 36% use a combination of lectures, textbook, and drawings • 81% use drawings and diagrams • All algorithm conceptualizations contained multiple snapshots to illustrate dynamic aspects of algorithm
(Lawrence, 1993, ch. 4.2)	Primary	35 students in a computer architecture class	To elicit student preferences for data representation of sorting elements	• Vertical and horizontal sticks were preferred to dots • Respondents requested labels
(Lawrence, 1993, ch. 4.3)	Primary	26 undergraduate CS students	To have students rank, in order of preference, 21 alternative representations of sorting elements	• Hollow, labeled, vertical bars were preferred to other representations
(Stasko, 1996)	Primary	Students in two algorithms courses	To elicit student opinions on algorithm animation assignments with Samba	• Student opinions regarding animation assignments were universally positive
(Stasko, Badre, & Lewis, 1993)	Complementary	10 participants in the experiment's animation group	To elicit student opinions on the animation they explored in the experiment	• All participants believed that the animation helped them • Participants requested complementary explanations • Participants wanted a way to rewind and replay the animation
(Reimer, 1996)	Complementary	All 6 study participants	To elicit participants' comments on their experience with the animations	• All participants enjoyed the animations, and felt that the animations assisted them in learning the algorithm

Table 7. Summary of the use of surveys and questionnaires in empirical studies of SV

an attempt to *control for* participant experience and ability. Second, in two of the studies, questionnaire data are explicitly analyzed and included in the results of the study.

Four empirical studies, in contrast, consider survey and questionnaire responses as their *primary* data. In these studies, all findings are drawn directly from participant responses, which, in some cases, are statistically analyzed (see Lawrence, 1993, ch. 4.1 and 4.2).

Table 7 summarizes the use of questionnaire and survey data in SV empirical studies, including the data's role in the study (*primary* or *complementary*); the study's questionnaire or survey respondents; the reason the questionnaire or survey are used in the study; and the findings the study draws from the questionnaire or survey data. As the table indicates, all of the questionnaires and surveys focus on the use of algorithm animation in educational settings. Half of them use questionnaires and surveys to elicit students' opinions on algorithm animation. Among that half, one result appears unanimous: *Students enjoy using algorithm animation, and they believe that it helps them learn about algorithms.*

Of the remaining three studies, two of them examine student preferences regarding data element representation and labeling in sorting algorithms. As reported above, Lawrence later finds (1993, ch. 4.4) that such preferences do not affect post-test scores. Finally, as part of their exploratory study that informed Lawrence's series of experiments Badre *et al.* (1991) learn that a large majority of professors appear to make use of drawings and diagrams in their teaching, and that sequences of multiple drawings and diagrams are used to convey algorithm dynamics.

3.4.4 Ethnographic field techniques

Distinguished by their commitment to collecting data in naturalistic settings, *ethnographic field techniques* include any of the qualitative techniques one might use to conduct a field study (Sanjek,

1995).[19] Perhaps the most eminent of these techniques, *participant observation*, is predicated on the idea that one can best gain an insider's view on the setting of interest by actually participating in it as an accepted member. Instead of merely observing members of the setting as they go about their business, the participant observer gradually gains the acceptance of her informants, allowing her to take on an increasingly participatory role in the activities of the setting. While an exhaustive list of ethnographic field techniques is beyond the scope of this article[20], other widely-used ethnographic field techniques include interviewing[21], artifact collection, diary keeping, and fieldnotes.

Three studies of SV use ethnographic field techniques for complementary data, while a fourth uses them as its main source of data. Both Badre *et al.* (1991) and Price (1990) use *exit interviews* in essentially the same way as other studies use exit questionnaires: to elicit participants' comments regarding their experiences in the study. Concurring with participants who responded to questionnaires and surveys in other studies of SV, participants in both of these studies said in interviews that they believed that animation had enhanced their understanding of the algorithm in some way. Ford (1993), in contrast, interviewed his participants in order to gain an understanding of the visualizations they had designed during the course of his study (see Table 4 for a summary of his results).

Finally, in the only study to rely exclusively on data collected by way of ethnographic field techniques, Bellamy (1994) uses both artifact collection and interviews to explore the ways in which experienced programmers use informal notations ("pseudocode," as she calls them) in their day-to-day programming activities. She finds that programmers naturally make use of graphical representations in at least four programming tasks: problem exploration, algorithm design, data structure design, and hand-simulation of program execution. In these tasks, participants use graphical notations for at least one of four purposes:

1. to make salient information available in an easily accessible visual form;
2. to express emerging solutions to design problems in alternative ways, which may accord with their initial conceptualizations more closely than formal programming language notations;
3. to assist in communicating with colleagues about design solutions; and
4. to explore, with a low level of commitment, alternative problem solutions.

According to the results of Bellamy's study, then, programmers customarily use pen-and-paper SV throughout the task of programming.

3.4.5 Usability tests

A special kind of observational study, the *usability test*[22] endeavors to identify and diagnose problems with an interactive system's user interface. In a usability test, researchers give a small number of participants, who may work in pairs[23] (usually three to five individuals or pairs in total), a set of tasks to perform with the system under study. The tasks are chosen so as to engage participants in scenarios that the system's designers believe the system should be able to handle. All interaction between participants and system is captured on videotape; in addition, the researchers typically take detailed notes during participant sessions. By reviewing the videotape and their notes, researchers pinpoint *breakdowns* in human-system interaction that may indicate problems with the user inter-

[19]The term *ethnography* has a double meaning, being both a collection of research techniques, and a genre of reportage (Sanjek, 1995). Thus, my decision to use the term *ethnographic field techniques*, as opposed to *ethnography*, is quite deliberate.

[20]See (Wolcott, 1992) for a taxonomy of ethnographic field techniques.

[21]See my justification for classifying interviews as an ethnographic field technique in footnote 18. See Spradley (1979) for an overview of the many ethnographic interviewing techniques, including structured, semi-structured, and casual.

[22]See (Hix & Hartson, 1993) for a more extensive treatment.

[23]The technique of having participants work in pairs is known as *constructive interaction* (Miyake, 1986); Douglas (1995) considers its advantages over single-participant usability studies.

face. By determining the cause of those breakdowns, they may be able to suggest ways of changing the interface such that users will not encounter the problems in the future.

Especially within industry, it seems likely that numerous usability tests have focused on SV features or systems. In most cases, however, those who ran the usability studies probably had little or no incentive to publish them. For one, it is difficult to imagine how a list of usability problems peculiar to one system would be relevant to other systems. Furthermore, at least in industry, it may be foolish to publish usability studies of products that are to go to market soon. Clearly, the results of such studies could run the risk of jeopardizing their image.

Perhaps because they ask research questions that transcended the usability of a specific software system, at least three usability tests involving tasks with SV systems can be found in the literature. To tune the Paradocs SV system for a controlled experiment (see Section 3.4.1 above), Price (1990) ran eight participants through a pilot usability test. In work related to the bubble sort observational study summarized in Section 3.4.2 (Douglas *et al.* , 1995, 1996), Hundhausen (1993) explores the use of constructive interaction and conversation analysis (as described by Douglas, 1995) in the study of human-SV system interaction. Finally, Lavery and Cockton (1995) are interested in comparing the ability of analytic and empirical evaluation techniques to identify usability problems with two prototype SV systems. In both studies, the authors demonstrate the efficacy of usability testing in diagnosing usability problems with interactive SV software. In the Lavery and Cockton's study, in fact, the authors find that their usability studies uncovered over twice as many problems as they found analytically.

3.5 Summary

The review of this section has outlined six alternative ways of studying SV tasks. As we have seen, by far the most effectiveness assessments have relied on programmatic and anecdotal evidence to substantiate effectiveness, with empirical and analytic techniques lagging far behind. The empirical studies of SV tasks have employed six research alternative research techniques—controlled experiments, observational studies, questionnaires and surveys, ethnographic field techniques, and usability studies. Most of the empirical studies have either worked within the experimental paradigm, or employed a less rigorous observational technique.

This section has additionally provided a comprehensive review of the empirical studies of SV tasks. The eleven controlled experiments have proposed a similar operational definition of effectiveness, and employed a uniform between-subjects design. The results have been mixed, with a majority of the studies failing to obtain the statistically significant result for which their authors had hoped. In stark contrast, questionnaire and survey data have painted an overwhelmingly positive picture of SV effectiveness; most people who use SV seem to think that it helps them. In an attempt to provide descriptive accounts of the ways in which, and the reasons for which, SV is used, observational and ethnographic studies have studied a variety of SV tasks. The accounts have been varied, emphasizing the diversity of ways in which SV is used and not used. Finally, although they are an established means of eliminating usability problems from an interactive system, published usability tests of SV systems have been few and far between, reflecting the reality that they may not be of interest to a wider audience, and that their publication may be a conflict of interest for their authors.

4. Analysis

The foregoing data and observations suggest that studies of SV effectiveness differ in at least three key respects:

- *their theories of effectiveness*—the ways in which they believe that the use of SV technology is beneficial;

- *their research techniques*—the ways in which they go about studying effectiveness; and
- *their research foci*—the research questions, people, SV systems, algorithms, and SV tasks on which they focus.

As a point of departure for the ensuing analysis, I suggest that these dimensions of differentiation embrace the three most important choices that researchers must make in their studies of effectiveness. By examining the consequences of choosing alternative theories of effectiveness, research techniques, and research foci, the analysis aims not only to paint a high-level picture of our current understanding of SV effectiveness, but also to critique the alternative accounts of effectiveness to which these different choices lead.

In turn, Sections 4.1, 4.2, and 4.3 take up each of the three key dimensions of differentiation. In addition, because all SV effectiveness studies share the common goal of being applicable to SV in practice, it is essential that this analysis also consider the extent to which such studies apply to the real world. A critique of the SV studies' *ecological validity* thus rounds out the analysis, setting the stage for the meta-study's synthesis in Section 5.

4.1 Theories of effectiveness

All of the research presented as data for this study either holds, or is interested in developing, a *theory of effectiveness*. In the case of a controlled experiment, for example, the experiment's theory of effectiveness is bound up in the particulars of the experiment's design, including its dependent and independent variables, procedure, participants, and materials. Independent variables indicate what the experimenters see as the *cause* of effectiveness; dependent variables indicate the experimenters' definition, or *operationalization*, of effectiveness; and the procedure, participants, and materials make a statement about the conditions under which such effectiveness is possible.

In less rigorous studies, theories of effectiveness may be provisional and evolving; hence, they may be more difficult to track down. Nonetheless, the research questions posed by an exploratory study usually suggest a theory of effectiveness. Consider, for example, Kehoe and Stasko's (1996) observational study of the use of SV in informal problem-solving sessions. The study poses the question of how and when alternative media are used in learning. One can infer, based on that question, that Kehoe and Stasko believe that not all media (e.g., text, animation, pictures) are created equal; certain media, they surmise, will be more *effective* than others for certain kinds of activities.

That a theory of effectiveness underlies any effectiveness study raises a crucial question: Is there a link between the theory of effectiveness assumed by a study, and the study's results? Especially for designers of controlled experiments in particular, an answer to that question is of great interest, since the success of controlled experiments hinges on statistically-significant results. Concurring with Kehoe and Stasko (1996), the analysis in this section takes as its point of departure the assumption that *a theory of effectiveness can make or break a successful effectiveness-validating effort*. I shall begin by introducing the concept of *division of labor*, which can be used to draw out the theories of effectiveness assumed by effectiveness studies. I shall then show that, in the case of controlled experiments, different divisions of labor have led to quantitatively different results. In particular, those experiments in which learners have been actively involved in the learning or problem-solving process have yielded more positive results.

4.1.1 Division of labor

While most studies of SV effectiveness may focus on a single SV task, all or most of the six tasks of SV must be done at some point prior to or during the study. Just who performs each task determines a *division of labor* for the study. To illustrate the concept, Table 8 presents the division of labor for the controlled experiment of Stasko *et al.* (1993). In that study, the SV tasks of Programming (in this case, programming of the pairing heap data structure), Data and Event Selection (the pairing heap's key operations), and View Writing (the actual animation used in the study) were all per-

formed by researchers (R) prior to the study. Within the study, participants (P) engaged in the tasks of View Exploration and Input Data Selection. In particular, they both *controlled* the animation through XTango's user interface, and chose the data and operations that drove the animation. Since they had control over the animation, the subscript *a* appears next to P in the VE column to indicate *active* exploration; contrast this with the *passive* exploration (denoted by a *p* subscript below) imposed on participants of Lawrence's (1993, ch. 4 and 5) initial experiments, in which participants simply watched animations. See Table 8's caption for a complete explanation of the notation.

P	DES	VW	VE	IDS	N
R	R	R	P_a	P	—

Table 8. The *division of labor* in the experiment of Stasko *et al.* (1993). The column headers P, DES, VW, VE, IDS, and N denote the six tasks of SV outlined in Section 2 (Programming, Data/Event Selection, View Writing, View Exploration, Input Data Selection, and Narration). Row entries indicate who, within the study, performed the SV task (R = the Researcher(s) who conducted the study, P = Study Participants, S = predefined by SV System, — = this task not relevant to the study). In the VE column, subscripts indicate whether view exploration was active (viewer had control of the visualization's user interface) or passive (viewer simply watched the visualization).

4.1.2 A division of labor analysis of the controlled experiments of SV

Table 9 presents the divisions of labor and results of all eleven controlled experiments. The divisions of labor are expressed in terms of the notation just described; an S (significant), NS* (significant, but the trend was in the right direction), or SW (significant in the wrong direction), together with a brief sentence, summarize each result.

STUDY	P	DES	VW	VE	IDS	N	RESULT
(Price, 1990)	R	S	S	P_a	R	—	**NS*** (Post-hoc analysis revealed that more in the animation group were "close" than in the control group)
(Stasko, Badre, & Lewis, 1993)	R	R	R	P_a	P	—	**NS*** (Non-significant trend favoring the text-and-animation group)
(Lawrence, 1993, ch. 4)	R	R	R	P_p	R	—	**NS** (Data representation does not matter)
(Lawrence, 1993, ch. 5)	R	R	R	P_p	R	—	**NS** (Data labeling does not matter)
(Lawrence, 1993, ch. 6)	R	R	R	P_a	P/R	—	**S** (Learner IDS is significant)
(Lawrence, 1993, ch. 7)	R	R	R	P_a	P	—	**S** (Conceptual. Step labeling and non-color algorithm operation labeling are significant)
(Lawrence, 1993, ch. 8)	R	R	R	P_a	P	—	**NS** (Text/animation order does not matter)
(Lawrence, 1993, ch. 9)	R	R	R	P_a	P/R	—	**S** (Lecture + lab with own learner IDS better than lecture only)
(Byrne, Catrambone, & Stasko, 1996, §2)	R	R	R	P_{a+}/P_p	R	—	**S** (Prediction and/or animation outperforms no animation/no prediction)
(Byrne, Catrambone, & Stasko, 1996, §3)	R	R	R	P_{a+}/P_p	R	—	**S** (Prediction and/or animation outperforms no animation/no prediction)
(Mulholland, in press)	R	S	S	P_a	R	—	**SW** (Visual tracer adversely affects problem-solving performance)

Table 9. The *division of labor* in all SV controlled experiments, along with their results. The division of labor notation is as described in Table 8**.** The RESULT column indicates the nature of the result obtained by the study; NS stands for Non-Significant, S stands for Significant, and SW stands for Significant in the Wrong Direction.

An elementary analysis of Table 9 indicates a two noteworthy trends. First, notice that both of the experiments in which View Exploration was passive yielded non-significant results. In other words, according to the researchers' operationalization of learning, it appears that students will not learn an algorithm by passively watching an animation.

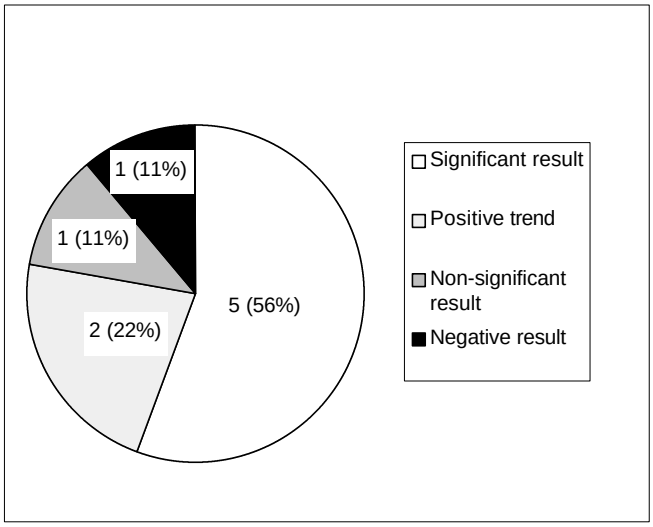

Figure 7. Summary of results of experiments with active viewer involvement

Second, observe that all four experiments that explicitly manipulate either View Exploration (active vs. passive) (Byrne *et al.*, 1996) or Input Data Selection (selected by student vs. selected by researcher) (Lawrence, 1993, ch. 6 and 9) produced a significant result. Clearly, allowing viewers to design their own input data (a P in the IDS column) and having students explicitly predict animation steps (a P_a in the VE column) can both be seen as ways of involving viewers more actively in the View Exploration process.

Third, of the five experiments in which View Exploration was active, but which do not manipulate that factor, just one (Lawrence, 1993, ch. 7) produced a significant result. Of the remaining four, two yielded a non-significant but positive trend, one produced a non-significant result, and one produced a negative result.

Figure 7 summarizes the analysis of experiments with active viewer involvement further. As it illustrates, seven of the nine experiments (78%) in which viewers were actively involved in the View Exploration task, produced "positive" results. Of the seven positive results, five of them are statistically significant, and two of them indicate positive trends. Seen in this light, viewer involvement, whether it be in the form of viewer Input Data Selection or active View Exploration, thus appears to be the single most important factor in the controlled experiments of SV.

LEARNING PARADIGM	SUPPORTING STUDIES	ONTOLOGICAL STATUS OF SV	THEORY OF EFFECTIVENESS
Passive viewer	(Badre *et al.*, 1991; Lawrence, 1993, ch. 4, 5)	Expert who is always right, and who knows exactly what examples to show	SV facilitates efficient, robust, and complete knowledge transfer
Active viewer/ problem solver	(Stasko *et al.*, 1993; Lawrence, 1993, ch. 6, 7, 8, 9; Reimer, 1996)	Expert who is always right, but whose illustrative examples are best guided by student interests	SV facilitates efficient, robust, and complete knowledge transfer, but learner must be involved in order to reap the benefits
	(Price, 1990; Mulholland, in press; Goldenson & Wang, 1991; Wilson *et al.*, 1995)	Problem-solving resource or environment	SV perspicuously presents the information needed to solve the problem
	(Kehoe & Stasko, 1996)	One of many learning media for self-guided problem-solving	SV provides useful medium for exploring procedural behavior of algorithm
Active viewer with explicit encouragement	(Byrne *et al.*, 1996)	Aid for teacher-guided structured drills	SV perspicuously presents salient information, and encourages *prediction* and *self-explanation* of procedural steps of an algorithm
Visualization Constructor	(Stasko, 1996)	self-reflective learning *process*	Process of building a SV forces students to become teachers, and thereby to learn algorithm

Table 10. Evolution of effectiveness theories in empirical studies of learning and problem solving with SV

Table 10 accounts for the historical evolution of SV learning studies (including observational studies of learning) in terms of the division of labor concepts introduced above. In two of the earliest studies, Badre *et al.* (1991) and Lawrence (1993, ch. 4 and 5) established the *passive viewer* paradigm, assuming viewer involvement in the View Exploration process to be inconsequential. Lawrence's next experiment (1993, ch. 6), however, marked a definite turning point, for it yielded a significant result by explicitly manipulating the person performing Input Data Selection; the viability of the *active viewer/problem solver* paradigm was thus established. In fact, from that point on, every experiment has maintained active viewer involvement.

P	DES	VW	VE	IDS	N
R	P	P	P_a	P	—

Table 11. The division of labor for the algorithm animation assignments proposed by Stasko (1996).

The most recent controlled experiments push the active envelope even further. In their experiments, Byrne *et al.* (1996) establish the *active viewer with explicit encouragement* paradigm, and substantiate its value in learning. Finally, in what might be characterized as a radical departure from the three viewing paradigms, Stasko (1996) advocates a *visualization constructor* paradigm, in which the learner explores algorithms by engaging in nearly all of the tasks of SV—from Data and Event Selection to View Exploration (see Table 11).

As Table 10 additionally illustrates, one can account for the evolving paradigms both in terms of the different *ontological status* that they ascribe to SV, and in terms of their differing theories of effectiveness. Whereas the earliest controlled experiments viewed SV learning sessions as didactic lectures in which the SV was an expert, the most recent experiments shift the ontological status of SV significantly, perceiving it not as an expert, but rather as a *tool* for interactive drills, or as a self-reflective *learning process*. Similarly, the epistemological foundation of theories of effectiveness has evolved from a pure Representationalist view of knowledge transfer (the proverbial conduit model described by Reddy, 1977), to a view of learning in which learners are more actively involved in constructing their understanding through prediction, self-explanation, and even through the entire process of SV programming (as in, e.g., Papert, 1980)

4.2 Research techniques

Given the tradeoffs among the differing theories of effectiveness posited by SV studies, one would expect to find similar tradeoffs among the alternative research techniques that advocate those theories. What alternative techniques have been enlisted in the study of SV effectiveness? How have the results varied? In addressing those questions, the analysis of this section shall illustrate (a) that one's research technique profoundly influences the kinds of effectiveness statements one ultimately makes, and (b) that choosing a research technique is largely a matter of finding an appropriate match to one's research questions.

By choosing to organize my review of the research into SV effectiveness (the data and observations of Section 3) around a taxonomy of research techniques, I have already performed at least part of the analysis of this section.[24] As that review indicates, all of the SV effectiveness research has employed one or more of four general techniques (revisit Figure 3, p. 3): *anecdotal*, *programmatic*, *analytic*, and *empirical*. Furthermore, the review illustrates an overwhelming bias toward the use of anecdotal and programmatic techniques; literature sources based on those techniques outnumber literature sources based on analytic and empirical techniques by over five to one (revisit Figure 4, p. 8). Of the empirical studies considered by the review, a plurality are controlled experiments. The usage of observational studies falls close behind, with somewhat lighter usage of questionnaires and surveys, ethnographic field techniques, and usability studies (revisit Figure 5, p. 12).

The following two subsections carry the analysis further—first, by examining the ways in which research techniques shape statements about SV effectiveness, and second, by considering the problem of matching research questions to research techniques.

4.2.1 Research techniques and statements about SV effectiveness

Table 12 juxtaposes the research techniques depicted in Figure 3 (p. 3) along several important dimensions: their units of analysis, methods used to collect and analyze data, and the kinds of results they aim for. In addition, the table's final column includes the findings of the previous subsection's analysis, indicating the theory or theories of effectiveness that can be at least loosely associated with each research technique.

Table 12 highlights three key considerations—*unit of analysis*, *data collection and analysis method*, and *desired results*—that profoundly influence the nature of the effectiveness statements made by studies employing those techniques. The DESIRED RESULTS column provides a logical starting point

[24]This observation reinforces my earlier caveat (Footnote 11) that one should be wary of any literature review that claims to be objective or neutral, since any literature review must necessarily assume some sort of analytical perspective. Indeed, how can one talk about such a large body of research without organizing it in some way?

TECHNIQUE	UNIT OF ANALYSIS	DATA COLLECTION METHOD	ANALYSIS METHOD	DESIRED RESULTS	THEORY OF EFFECTIVENESS
Anecdotal Evaluation	A memory or experience of SV system use	No data, just recollections	Self-reflection and writing	Persuasive rhetorical argument for technology feasibility and usefulness	Spot examples of an SV system's successful application to real or plausible problems illustrate the system's effectiveness
Programmatic Evaluation	Source code for a SV	Gather visualization programs that produce visualizations of interest	Count lines of code or number of rules, or simply present program	Few number of lines or rules, or readable code	Effective visualizations are generated by short, readable visualization programs, which must be easy to write
Analytic Evaluation	SV system, possibly with respect to a set of tasks	System use and note-taking	Walk through tasks with critical eye, or heuristically apply principles of design to interface	List of potential usability problems, along with possible solutions	Effective SV systems adhere to principles of good design, and are devoid of usability problems
Controlled experiments	Individual knowledge or behavior	Measurement of predefined behavioral observables, such as test scores and test-taking time	Statistical tests	Statistically-significant differences that favor SV technology	Varies from study to study; see Table 10.
Observational studies	Individuals or groups engaged in SV tasks	Videotaping, interviews, questionnaires, surveys	Protocol analysis, Conversation analysis, Interaction analysis, and others	Qualitative accounts of processes by which humans engage in observed tasks, ideas for future research	Varies from study to study; see Table 10 for summary of learning studies
Questionnaires and surveys	Individuals	Written questionnaires and surveys	Counts, catalogs, statistical tests	Quantitative or qualitative accounts of individual preferences or impressions; often used to complement other methods	Varies from survey to survey; in some cases, the survey aims to develop such a theory
Ethnographic field techniques	Recurrent social scenes	Field notes, participant observation, interviews, videotaping, artifact collection, diary keeping, and others	Interview transcription and analysis, field note consolidation, to name but a few	Qualitative descriptions of shared cultural knowledge and practices from the perspective of an insider.	None—may be used to develop such a theory based on users' authentic activities
Usability studies	Individuals or group performing tasks with a specific SV system	Videotaping, questionnaires, interviews	Same as for observational studies	List of usability problems, along with possible design solutions	Effective SV systems are devoid of usability problems; they enable users to perform tasks non-problematically

Table 12. Comparison of the six approaches to studying effectiveness considered in the review of Section 3.

for understanding the differences in effectiveness statements made by studies employing each technique. On the one extreme, anecdotal techniques aim for nothing more than *persuasive arguments* in support of SV technology. On the other extreme, controlled experiments enlist the Scientific Method in an attempt to establish a *causal link* between SV technology and effectiveness. Between these two extremes, one finds a range of desired outcomes, each of which has something qualitatively different to say about SV effectiveness, or about the conditions that might lead to it. Questionnaires and surveys, for instance, seek *individual preferences* or *opinions*, which may reflect people's subjective experiences with SV technology.

In light of the range of different kinds of results for which the research techniques strive, it should come as no surprise that they recommend a corresponding range of data collection and analysis techniques. For example, in order to assert causality, controlled experiments must adhere to the stringent requirements of the experimental methodology upon which they are based. In particular, controlled experiments must operationalize the behavior of interest *a priori*, impose tight environmental controls, and sample data in accordance with the assumptions of statistical models. By contrast, observational studies are interested in producing qualitative accounts whose plausibility is firmly grounded in empirical data. As a result, they enlist a markedly different data collection and analysis techniques, which may be every bit as systematic as those employed by controlled experiments (see, e.g., Jordan & Henderson, 1994), but which focus on videotaped episodes instead of performance measures.

The techniques' units of analysis depend more on the *intellectual tradition* (Sanderson & Fischer, 1994) or community of practice with which they are associated than on the form of their desired results. First, the Behavioral tradition within which controlled experiments are conducted, as well as the Cognitivist tradition[25] out of which usability studies, protocol analysis-based (see Ericcson & Simon, 1984) observational studies, and analytic techniques have evolved, has a longstanding interest in *individual* cognition. As a consequence, those who conduct controlled experiments tend to operationalize and measure *individual* behavior, and those who perform analytical evaluation, usability studies, and protocol analysis-based observational studies tend to focus on *individuals* performing tasks.[26] Second, many observational techniques, as well as ethnographic field techniques, have evolved out of the Social tradition, whose roots lie in sociology and anthropology. As a result, their units of analysis tend to involve groups of interacting people. Finally, with their origins in the software engineering community, programmatic techniques focus on *computer programs*.

4.2.2 Matching techniques to effectiveness questions

That each research technique produces effectiveness statements of a different flavor implies that different techniques will be appropriate for different types of research questions. Selecting an appropriate technique thus involves finding a match between one's research questions and a technique that can provide satisfactory answers to those questions. To provide some guidance, Table 13

[25]I use both *Behaviorist tradition* and *Cognitivist tradition* strictly in the senses in which they are used in (Sanderson & Fischer, 1994).

[26]However, note that, in order to create a more natural social situation, some researchers opt to employ groups of participants in usability studies; see Footnote 23 (p. 21).

TECHNIQUE	APPROPRIATE RESEARCH QUESTIONS	ADVANTAGES	DISADVANTAGES
Analytic Techniques	What are the usability problems with an SV system, and how might we fix them?	• Can be performed quickly	• May miss many of the problems found in usability tests
Controlled Experiments	Does some SV technology factor cause some effect, where that effect can be precisely stated in terms of measurable and observable phenomena?	• Results are quantitative and thus seen as hard evidence • Results have high degree of objectivity, since they are based on pre-determined observable and measurable variables • Experiments can be replicated, which lends to their credibility • If requirements of statistical models are met, results may be generalized to population	• Studies require large sample sizes to meet the assumptions of the statistical models • Participants must be carefully screened; it may be difficult to find study participants who meet the selection criteria. • It is difficult to control for all possible factors, and hence to assert causality • Difficult to ensure that experimental conditions have equal access to equivalent information • Controlled experimental settings reduce ecological validity
Observational Studies	How do humans engage in an SV task? What resources do they make use of, and how?	• Can provide fine-grained accounts of how humans engage in SV tasks • Studies can be conducted with a small number of participants • Can generate research questions and hypothesis for future study • Well-suited to study of collaborative tasks	• Difficult to generalize human behavior from small number of participants • Qualitative results may appear soft, may not be regarded as hard evidence • Since results rely on post-hoc analysis, objectivity may appear damaged
Questionnaires and Surveys	What are user preferences with respect to a particular visualization or visualization system? What do they wish they had had? What would they like to see in future versions?	• Can guarantee anonymity • Unintrusive, especially if administered via e-mail • Efficient means of obtaining complementary subjective data on participants' preferences and opinions in an empirical study	• Sometimes difficult to know which questions to ask • Difficult to know whether respondents provided answers that are indicative of their true opinions, or whether they provided answers they believed the researcher wanted.
Ethnographic field techniques	How might SV technology fit in to the overall practices of a cultural scene? What knowledge about an SV artifact is shared by members of cultural scene?	• In the early stages of research, can help to determine ways in which SV technology might fit into the activities in a cultural scene • High degree of ecological validity	• May be difficult, from a practical standpoint, to arrange fieldwork • May be time-consuming; in typical ethnographic fieldwork, one needs to conduct a minimum of 2-3 months of fieldwork to "get into" culture • Unit of analysis too broad to provide insight into detailed human-visualization interaction
Usability Studies	What are the usability problems with an SV system, and how might we fix them?	• Effective means of evaluating the user interface of an interactive system • Three to five participants sufficient to identify most of a system's problems	• Narrow scope: Does not consider questions beyond those at the level of tasks and the user interface • Can be difficult to find design solutions to fix problems

Table 13. The kinds of research questions that the six approaches are designed to answer, as well as some of the approaches' advantages and disadvantages

suggests a set of general SV effectiveness questions that each of the techniques is designed to address.[27] For instance, the table indicates that, to identify and diagnose the usability problems in a prototype SV system, one should choose between *analytic techniques* and *usability tests*.

Answers to the research questions listed in the second column of the table do not come without a price; associated with any research technique is a set of pragmatic and methodological tradeoffs to be considered. Columns 2 and 3 of the table list some of the most important of these for each technique. For example, while the results of controlled experiments are generally perceived as hard evidence of effectiveness, it may be difficult, from a practical standpoint, to stage such a study, as Stasko *et al.* (1993) point out:

> Pragmatically, it is challenging to assemble the appropriate ingredients for a [controlled experiment]. . .[A] group of subjects who are at an appropriate point in their educational careers must be available. Even this may not be enough, however, because splitting the subjects into two groups, one using animation and one not using animation, may unfairly influence student achievement and grading the particular course in which the students are enrolled. (p. 61)

While observational studies can provide detailed accounts of the processes by which humans engage in SV tasks, small sample sizes limit the extent to which those accounts can be generalized. Questionnaires and surveys provide an ideal means of collecting subjective data; however, it may be difficult to determine whether respondents provided accurate answers, or answers they believed the researcher wanted. Ethnographic techniques are well-suited for obtaining an insider's perspective on the artifacts, activities, and beliefs of a particular cultural scene. To obtain such an insider's perspective, however, typically requires a relatively long time in the field—two to three months. Finally, while usability studies are effective for identifying usability problems with interactive SV systems, it may be difficult to glean solutions to those problems from a usability test. Further, their narrow scope limits them from addressing questions that lie beyond usability (e.g., usefulness).

4.3 Research Foci

Just as the research techniques employed by SV effectiveness studies have influenced the results they have yielded, so too has their choice of what to focus on. In this segment of the analysis, I scrutinize the research questions, people, SV artifacts, algorithms, and SV tasks that have been of interest to SV effectiveness studies. As I shall illustrate, most SV effectiveness studies have maintained a focus on View Exploration tasks in which undergraduate computer science students use a few SV systems to learn about textbook algorithms.

4.3.1 Research Questions

While the SV effectiveness studies have examined a wide variety of specific research questions relating to their unique circumstances, it is possible to unify many of those questions at a higher level of abstraction. Figure 8 presents a summary of the seven high-level research questions posed by the SV effectiveness studies. As the figure indicates, the general question of what factors influence learning with SV technology—posed by nearly all of the controlled experiments (Lawrence, 1993, ch. 5 – 9, Stasko *et al.*, 1993; Byrne *et al.*, 1996), and by one survey (Stasko, 1996)—has been of most interest. The overriding goal of the studies that ask this question is to furnish computer science educators with evidence in support of SV's pedagogical benefits. As Byrne *et al.* (1996) put it,

> [c]onstructing algorithm animations generally requires serious programming effort, and it still has to be demonstrated that the benefits justify this cost. If the same pedagogical advantages can be realized with less labor-intensive materials, then the less labor-intensive methods make more sense. It is incumbent upon educators and animation builders to carefully examine their assumptions about what students will learn from an animation, and why it is that an animation is necessary to convey the desired information. (p. 20)

[27] Because their plainly impoverished views of effectiveness, anecdotal and programmatic techniques are excluded from the table.

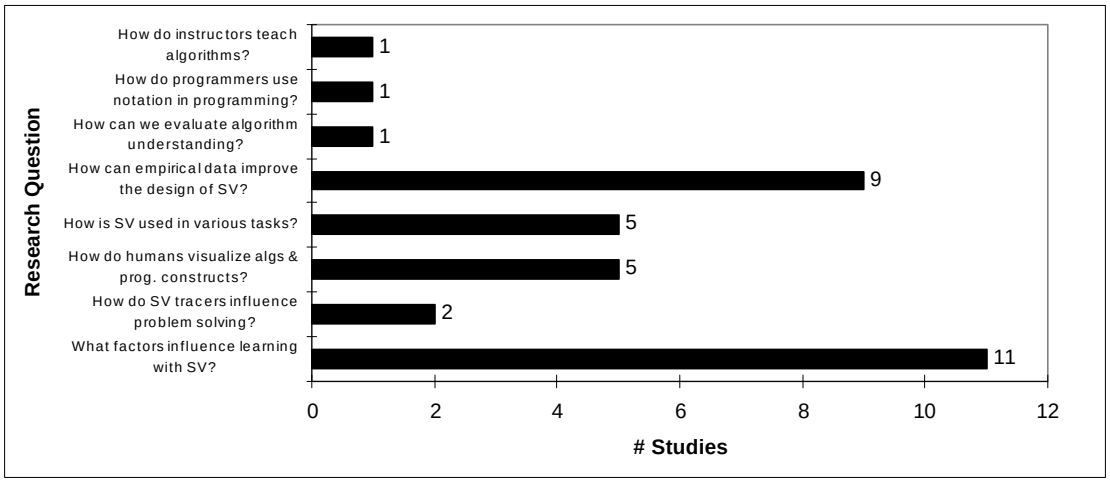

Figure 8. Summary of primary research questions explored by SV effectiveness studies.

A secondary concern of the studies that probe SV learning factors is to provide computer science educators with *guidelines* for the effective use of SV in educational settings. As Lawrence (1993) points out, "There are several possible approaches [to integrating SV into the computer science curriculum], ranging from classroom lecture examples to supervised laboratory presentations to unsupervised or discretionary use" (p. 2).

In response to the observation that most SV technology is based on armchair design, nine effectiveness studies probe the question of how empirical data can be used to inform the design of more effective SV technology (Lawrence, 1993, ch. 4.2 and 4.3; Hundhausen, 1993; Douglas *et al.*, 1995, 1996; Chaabouni, 1996; Reimer, 1996; Lavery & Cockton, 1996). These studies aim either to root SV technology design in techniques (e.g., user-centered design, usability testing) drawn from the emerging research on human-computer interaction (HCI), or to tailor such techniques to the particular needs of SV design.

While not focused as intently on actual design issues, studies that ask two related questions—How is SV used in various tasks? (Goldenson & Wang, 1991; Wilson *et al.*, 1995; Kehoe & Stasko, 1996) and How do humans visualize algorithms and programming constructs? (Badre *et al.*, 1991; Ford 1993; Douglas *et al.*, 1996)—may offer additional insights into empirically-based design. Indeed, these studies concern themselves with describing both the processes by which humans engage in various View Exploration tasks, and the forms that their visualizations take.

The two studies that pose the question of how and whether graphical Prolog tracers benefit program tracing tasks (Price, 1990; Mulholland, In press) might be viewed as analogs to the studies on SV learning factors. Like the SV learning factors studies, they are interested in establishing an empirical basis for the use of SV technology in a particular kind of task. The primary difference lies in the task of interest—program tracing, as opposed to algorithm learning.

Finally, two other questions reflect researchers' varying interests in designing controlled experiments (Badre *et al.*, 1991) and exploring the notations used by professional programmers in their day-to-day programming (Bellamy, 1994). The studies that ask these questions appear to have definite future work in mind—in the former case, controlled experiments investigating SV learning factors, and in the latter case, empirical comparisons of various tools' support for the programming task.

4.3.2 SV artifacts and target programs

Figure 9 provides a breakdown of the SV artifacts employed in the SV effectiveness studies. Because View Exploration has been the most widely studied task (see Section 4.3.3 below), the important issue in the analysis at hand is the *user interfaces*[28] through which users have explored visualizations with these artifacts.

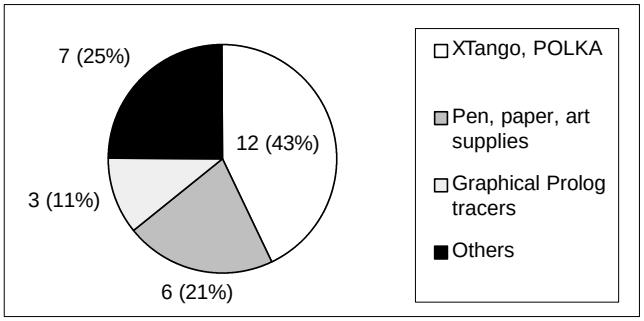

Figure 9. Summary of SV artifacts used in SV effectiveness studies

Both XTango and POLKA support a subset of the set of interface operations pioneered by Brown's original BALSA systems (Brown, 1988a). Using a standard control panel (see Figure 10), XTango and POLKA users can *play* and *pause* an animation, *adjust* its *speed*, *zoom* into and out of it, and *pan* around in it. While not much is reported about participants' experiences with XTango and POLKA in the *active viewer* studies (see Section 4.1) involving those systems, participants have commonly complained about the lack of a *rewind* operation (see, Stasko *et al.*, 1993), which would allow an active viewer to go back and review an animation at any point.[29] Within the context of a debugging task, Price's (1990) usability test confirms the importance of such a rewind function; indeed, his observations lead him to add the feature to the version of ParaDocs that he ultimately uses for his controlled experiment.

Perhaps a bit surprising is the finding that 21% of the studies abandon SV technology altogether, opting instead to explore SV with pen, paper, and common art supplies. Because it is widely held that the task of implementing visualizations within an SV system can be formidable [30], these studies aim to circumvent the difficulties of SV programming by allowing participants to create visualizations using media using artifacts with which they are presumably familiar and comfortable.

Finally, of the "Other" studies, two (Hundhausen, 1993; Lavery & Cockton, 1995) focus specifically on SV system usability, while a third closely examines students' use of a web browser into which algorithm animation is integrated (Kehoe & Stasko, 1996). In all three studies, along with Price's (1990) study cited above, SV system user interfaces play an important role in participants' successfully accomplishing tasks. All of these studies indicate, to varying extents, that negotiating a SV system's user interface, like interpreting an animation, can be a barrier to task success.

Figure 11 considers the target programs used in the SV effectiveness studies. By far the most frequently studied programs are the kinds of sorting, graph, and tree algorithms one could expect to learn in an undergraduate algorithms course. Given the studies' keen interest in the use of SV for computer science education, that observation should come as no surprise.

[28]See the *Interaction* dimension in Price et al.'s (1993) taxonomy for a synopsis of the issues involved.
[29]As Brown (1988*a*) notes, including such a rewind capability in *live* algorithm animations (as opposed to *post-mortem* animations) poses a formidable technical challenge; this may well explain its absence in XTango and POLKA.
[30]See, e.g., Brown & Sedgewick's (1985) anecdotal report on building visualizations in BALSA.

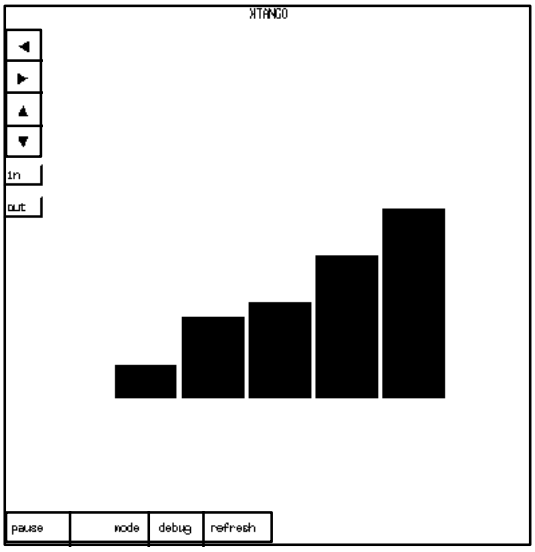

Figure 10. The standard *play-pause-zoom* interface offered by 43% of the SV systems studied

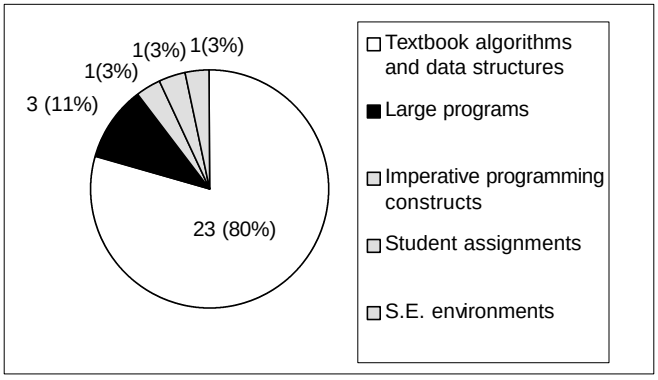

Figure 11. Summary of the target programs employed by participants in the SV effectiveness studies

Of the studies that do not focus on textbook algorithms, two stand out for their attempt to study larger-scale programs. While the size of the 7,500-line operating system simulator that Price's (1990) participants debugged may pale in comparison to typical industrial programs, it is certainly significantly larger than a textbook algorithm. The programs Bellamy (1994) considers may well be larger, for she examines the notations of professional programmers engaged in genuine programming projects in industry.

4.3.3 People and tasks

Turning to the kinds of people who participate in the SV effectiveness studies, we find that undergraduate computer science and engineering students are by far the most popular participants, with graduate students a somewhat distant second (Figure 12).[31] Interestingly, despite their central in-

[31]Much of the literature makes distinctions among students that are finer than those that appear in Figure 12. For example, the literature often refers to participants as "upper-level computer science students" (see, e.g., Byrne et al., 1996, §3) or "beginning computer science students enrolled in their first programming class" (see, e.g., Lawrence, 1993, Ch. 5). Due to the general lack of consistency in such fine-grained participant classifications, I have adopted the finest distinctions I could reliably make across studies In the case of computer science students, the best I could do was distinguish between undergraduates and graduates.

volvement in computer science education, professors have participated in just one study. Also notable is the lack of studies that consider SV tasks involving professional programmers.

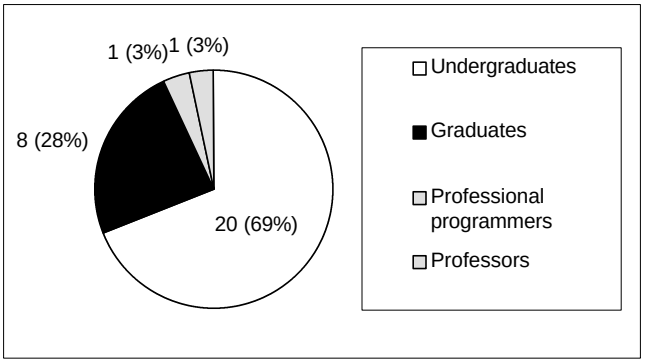

Figure 12. Summary of the kinds of people who have participated in SV effectiveness studies

Figure 13 depicts the extent to which the six SV tasks have been studied. Given that View Exploration is widely held to be the task in which one presumably reaps the benefits of SV, and given the studies' common interest in probing the effectiveness of SV in teaching textbook algorithms to computer science students, it should come as no surprise that View Exploration tasks have been the most frequently studied. Note that View Exploration tasks have sometimes included Input Data Selection—as, for example, in the case of the controlled experiments (Lawrence, 1993, ch. 6 – 9; Stasko *et al.*, 1993). In contrast, none of the View Exploration tasks in controlled experiments has involved Narration.[32]

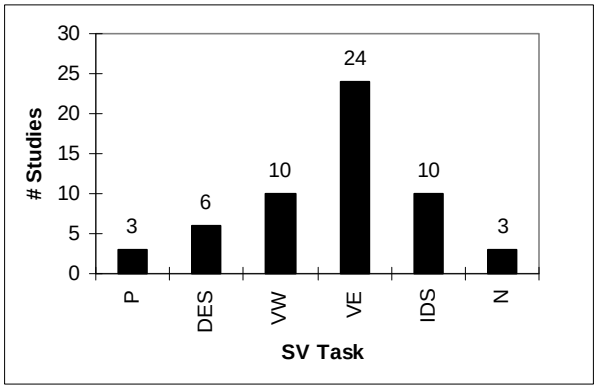

Figure 13. Comparison of the number of SV effectiveness studies that have considered the six tasks of SV. Study counts are further broken down by technique. Those studies that consider more than one task are counted multiple times—once for each task they consider.

Recall that the focus of a Visualization Exploration task is not the visualization itself, but rather the *goal* of the task in which the visualization is being enlisted (see Section 2.5). Figure 14 details the nature of the View Exploration tasks that have been studied. As the figure suggests, by far the most widely studied Visualization Exploration task is rather vague: *Learn the algorithm to prepare for a subsequent test.* At least in controlled experiments, the ambiguity of that task appears problematic, both for the experimenter, and for the learner.

[32]One might argue, however, that Byrne *et al.*'s (1996) prediction tasks involved a form of narration—namely, predicting the next snapshot of an animation. As I indicated in Section 4.1, for the purposes of this analysis, I consider such prediction to be form of *active viewing*, rather than narration. As defined in this meta-study, Narration fulfills a *communicative*, as oppsed to a *self-reflective*, role.

For the experimenter, such an ambiguous empirical task might lead to the design of experiments in which participants produce unexpectedly low test scores, as Stasko *et al.*'s (1993) experiment inadvertently demonstrates. In particular, they find that, given the instructional materials through which participants are to prepare for the test, they could not possibly have been expected to perform well on the test. Committing this mistake leads Stasko *et al.* to emphasize the point that instructional materials need to be firmly rooted in the ultimate material to be tested.

On the other hand, the lack of a well-defined task objective places learners in an unrealistic (and potentially disconcerting) situation in which they are to explore an animation without a clear idea of what they are supposed to get out of it. As learners engage in an aimless process of discovery learning, there is no guarantee that they will stumble upon the insights into algorithmic behavior that could help them on the upcoming test. Thus, the lack of a concrete objective may serve to rob an animation learning session of its putative benefits.[33]

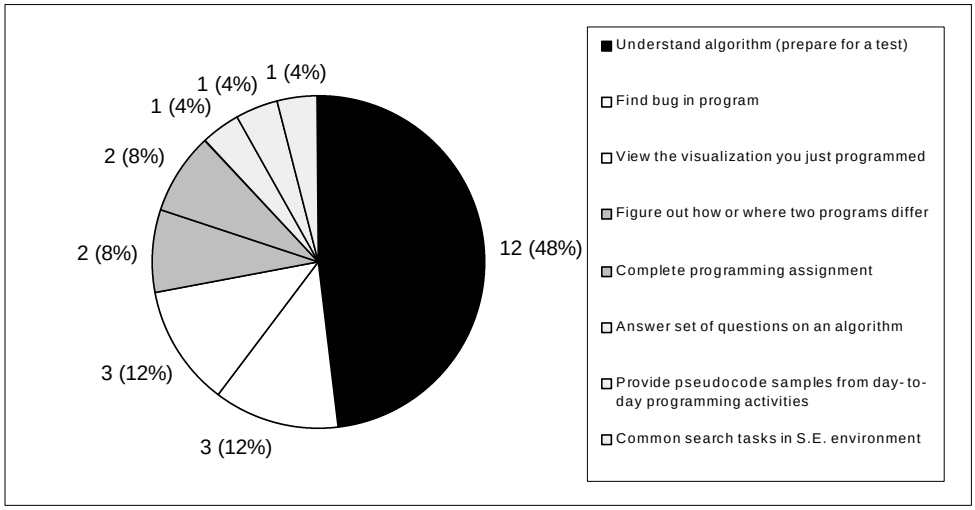

Figure 14. Breakdown of the Visualization Exploration tasks examined in SV effectiveness studies. The general task of understanding has been the most widely studied by far.

Further analysis of Figure 13 reveals that a secondary interest of the studies has been *visualization programming* (Data/Event Selection and View Writing). Besides confirming the difficulty of visualization programming, especially as compared to visualization construction with paper, pencil, and art supplies, three of these studies (Douglas *et al.*, 1995, 1996; Chaabouni, 1996) illustrate the essential role of *narration* in tasks in which visualization is subsequently used as a resource for explanation. Interestingly, using a radically different research technique, Stasko *et al.* (1993) arrive at the same conclusion:

> To be most effective, algorithm animations must be accompanied by comprehensive motivational instruction. The quality of these teacher-provided explanations *is perhaps even more important than* the animation itself (pp. 65-66, emphasis mine).

Given these findings, it is thus noteworthy that the design of current SV technology appears to neglect the importance of narration.

With respect to the tasks considered by the SV effectiveness studies, a final observation—one that delves below the surface of Figure 14—is the absence of studies that provide an integrated view of the interrelations among tasks within scenarios of SV use. As indicated in Section 2.7, in any sce-

[33]Notice that the observation that task ambiguity may purge SV of its benefits may account for Byrne *et al.*'s (1996) shift to the *active viewer with explicit encouragement* paradigm (see Table 10), which can be seen as an attempt to define the View Exploration task more precisely.

nario of SV use, the SV tasks of Programming, Data/Event Selection, and View Writing must all be completed prior to the View Exploration task. With regard to the 24 studies that examine View Exploration, one cannot help but wonder about the details of the other SV tasks (Programming, Data/Event Selection, View Writing) that necessarily compose the scenario of SV use in which the View Exploration takes place. Indeed, since most of those tasks end up being performed by the studies' researchers outside of the study, they are rarely analyzed. As a consequence, the studies fail to consider their implications for the tasks that are actually examined.

Yet, as most SV effectiveness researchers would readily admit, the difficulty of performing such tasks as Data/Event Selection and View Writing, as well as the interrelations among SV tasks, can significantly influence one's choice of whether, and how, to use SV in practical settings. Indeed, the real world embodies myriad constraints that an empirical study, whether it be controlled or observational, simply cannot anticipate, much less systematically analyze.

4.4 Are we studying what counts? Applicability to SV in the wild

In bringing to light the significant incongruities between written mathematics tests, and the practice of mathematics in the real world, Lave (1988) points out the dangers of assuming that the results one obtains under the closed conditions of the laboratory can speak to the authentic practices of people in the lived-in world.[34] Insofar as they take participants away from their everyday activity, calling upon those participants to engage in inauthentic SV tasks within inauthentic settings, the empirical studies considered by this study clearly run the risk of yielding results that simply do not speak to SV in practice. I conclude my analysis by considering the *ecological validity* of the SV effectiveness studies, in an attempt to gauge the extent to which they apply to SV in the wild.

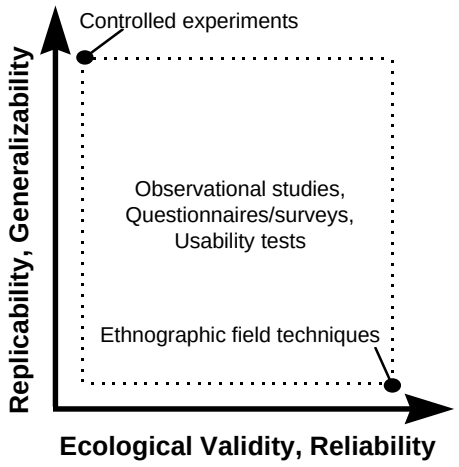

Figure 15. Mapping the SV research techniques onto the grid of replicability, generalizability, ecological validity, and reliability

As shown in Figure 15, based on their varying levels of interest in *validity*, *reliability*, *replicability*, and *generalizability*, the empirical research techniques taxonomized in Figure 3 can be placed on a two-dimensional grid. As one moves from left to right on the grid, the techniques impose fewer and

[34]Her extensive ethnographic study finds, in fact, that the shape of mathematics problems in the lived-in world of American adults does not even loosely correspond to the shape of the problems one finds on scholastic mathematical tests. In stark contrast to the latter, answers to problems in the former appear to emerge *before* the problems themselves, relegating one's problem solving activity to that of "closing the gap" to an emergent solution using the resources presently at hand. In perhaps the most famous of Lave's examples, a woman is trying to take three-quarters of a recipe that calls for two-thirds of a cup of cottage cheese. While grade school mathematics suggests that the solution to the problem is to apply fractional multiplication ($^2/_3 * ¾ = ½$), the woman instead "fill[s] a measuring cup two-thirds full of cheese, dump[s] it out on a cutting board, pat[s] it into a circle, mark[s] a cross on it, scoop[s] away one quadrant, and serve[s] the rest" (Lave, 1988).

fewer environmental controls; their ecological validity ("Are we studying activities in which people really engage?") and reliability ("Are we testing what we claim we are testing?") thus increase. As one moves from bottom to top on the grid, the techniques impose increasingly many environmental controls; their replicability ("Can someone else repeat this test and produce the same results we obtained?") and generalizability ("Do the results we obtain under the conditions of our experiment generalize to the entire population from which we sampled?") increase accordingly. Clearly, controlled experiments and ethnographic field techniques occupy the two extremes of the grid, with all other techniques falling somewhere in between.

Based solely on the preceding analysis, one might reasonably conclude that the lone study that relies exclusively on ethnographic field techniques (Bellamy, 1994) has the most ecological validity; that the eleven controlled experiments have the least ecological validity; and that the remaining sixteen studies fall somewhere in between. While perhaps a reasonable starting point, such an analysis is plainly impoverished, for it ignores both the details of the tasks investigated by the SV effectiveness studies, and (thus) the possible parallels between those tasks and SV in the real world.

To carry the analysis further requires the kinds of descriptive accounts of SV in practice that Bellamy's (1994) ethnographic enquirey provides. Unfortunately, aside from Bellamy's, no ethnographic accounts of SV use in the real world appear in the literature. Consequently, the remainder of this analysis comes with the caveat that it is plainly provisional, relying on anecdotal reports of SV use that appear in the literature, on an ethnographic field study in which I am presently engaging (Hundhausen, In preparation), and on my own experience with SV (mainly in academia) since 1988.

Figure 16 proposes a high-level taxonomy of SV use in the wild. While I can make no claim that it portrays the way SV is actually applied in practice, it might be characterized as the *normative* view of SV use, and I take comfort in the fact that a forthcoming book (Brown *et al.*, In press) dedicated to SV concurs with its top-level categories (*Education* and Software *Engineering*).

As the figure indicates, members of two major enterprises—academia and industry—enlist SV technology to fulfill markedly different high-level objectives. In education, the high-level goal of SV is to assist in *the training of competent software engineers who can contribute to society*. In industry, on the other hand, the high-level goal of SV is to assist in the *on-time development of marketable software*.

Within these two enterprises, people enlist SV in a variety of ways, owing to the specific social and organizational structures of the enterprises. The leaf nodes of the taxonomy distinguish such applications of SV based on their goals, which are distinct from, but respond to, the overall goal of SV in each respective enterprise.[35]

In a CS course, for example, several well-established *recurrent social scenes* (Spradley & McCurdy, 1972) distinguish various uses of SV technology:

- *Lectures*. In high school classrooms and college lectures, computer science instructors use graphic representations to help them explain aspects of the algorithms under study. The advent of graphical workstations in the early 1980s paved the way for the use of computer-based, interactive visualizations and animations in lectures, as pioneered by professors in Brown's electronic classroom (Brown, 1988a, Appendix A; Bazik, Tamassia, Reiss, & van Dam, In press). As Gurka and Citrin (1996) put it, SV in lectures is essentially "an extension of the blackboard, but with more capabilities available" (p. 183).

[35]It strikes me that the descriptive framework offered by activity theory (see, e.g., Bellamy, 1996) would serve this analysis well. However, an introduction to the concepts of activity theory is beyond the scope of this article.

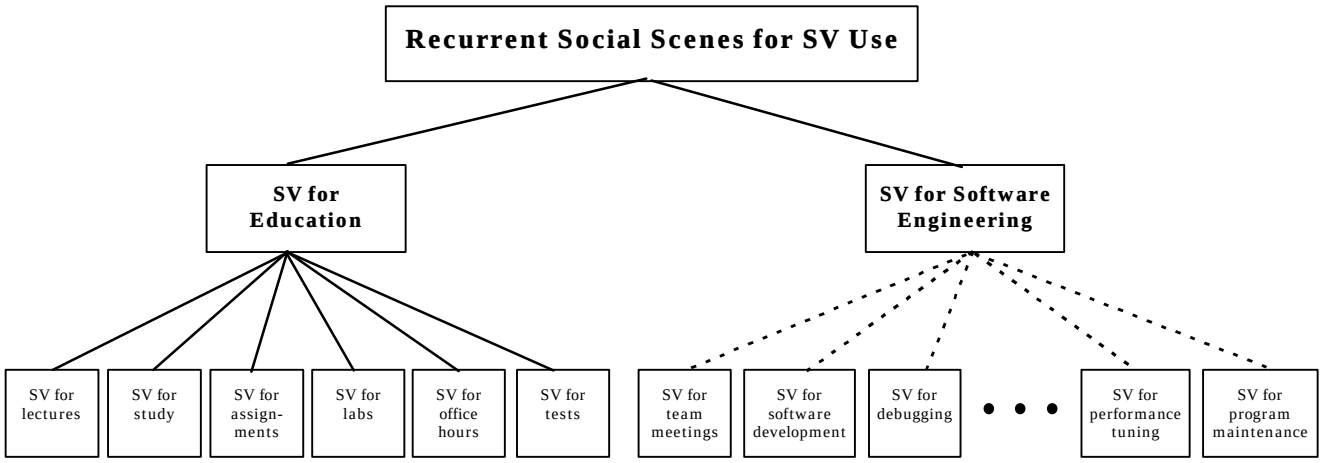

Figure 16. A provisional taxonomy of scenarios of SV use in practice. The dotted arcs adjacent to SV for Software Engineering indicate the highly speculative nature of the leaf nodes. The dots in between the SV for Software Engineering leaf nodes indicate that there are likely many more recurrent social scenes in software engineering, and that they probably vary between software teams. On the SV for Education side of the taxonomy, on the other hand, the recurrent social scenes are less speculative and probably vary less from class to class.

- *Assignments*. Students are free to work on course assignments on their own time, as long as they hand them before the established deadline. Two of the empirical studies considered in Section 3.4 suggest alternative uses of SV for assignments. First, Goldenson and Wang (1991) describe the ways in which students use the Pascal Genie programming environment, which has built-in design- and run-time SV tools, to complete course assignments. Second, Stasko (1996) advocates student assignments in which students develop their own animations of the algorithms under study.

- *Laboratories*. In algorithm visualization laboratories, students interactively explore algorithms and data structures through structured laboratory exercises (Naps, 1990). Like assignments, laboratory sessions have a concrete goal, embodied in a deliverable assignment. However, unlike assignments, labs are constrained by both a location (a laboratory containing graphic workstations) and a contiguous block of time (a session or sitting). Further, students in a laboratory are assured access to a knowledgeable teacher or lab assistant for the duration of the lab. Finally, whereas assignments often require students to delve into the grubby details of programming, including input and output, laboratory exercises are designed such that students only have to fill in key procedures and functions, thus giving them more time to explore the algorithms visually (Naps, Personal communication).

- *Study*. Students enrolled in computer science courses have the opportunity to *study* for tests at any time. Study differs from laboratories in two key respects. First, students need not study in a *single sitting*, but rather may study on their own time. Second, lacking a precisely-defined goal, study is more open-ended. Indeed, individual learning styles, preferences, and time constraints may dictate radically different approaches to study. Depending on their individual preferences, students may elect to enlist SV in their study by drawing their own visualizations, by examining hard copies of visualizations constructed by others (professors or book authors), or by using interactive SV software to which they have access.

- *Office Hours*. In college courses, professors and teaching assistants schedule weekly *office hours*, during which students in need of assistance may visit them. Since office hour sessions are often one-on-one, instructors can provide students with more personal attention than they can in lectures, and students may be less reluctant to ask questions. Instructors may use SV to help them diagnose bugs in students' programs (Gurka & Citrin, 1996), or to help

them answer student questions. In the latter case, SV plays a communicative or explanatory role, akin to its role in lectures.

- *Tests*. Like laboratories, course *tests* must be completed during a contiguous block of time. However, unlike both assignments and laboratories, tests are generally taken under *closed conditions*; students are denied access to outside study materials, forcing them to rely on their own memory and wits. In test-taking situations, SV can be used to help pose questions. For example, Brown (1988*a*, Appendix A) reports that exams in the algorithms courses at Brown often included stills of algorithm animations discussed in class; students would be asked to "name-that-algorithm," just as students in an art history class might be asked to identify paintings. Alternatively, a test question might have students indicate the behavior of an algorithm by drawing a series of data structure snapshots.

Similarly, within industry, the social and organizational characteristics of a software development team suggest alternative uses of SV.[36] Unfortunately, the relative paucity of literature on the use of SV in industry prevents me from sketching out any more than a speculative account of the different social scenes in which it is used.

- *Team meetings*. Bellamy (1994) finds that a primary use of visual notations of programs is in communicating with colleagues. In such settings, visualizations allow "ideas about a program that are not yet fully formulated to be shared with others" (p. 235).

- *Software development*. As programmers design and implement code, they make extensive use of graphical notations (Bellamy, 1994). In addition, programmers may use computer-based programming environments, such as FIELD (Reiss, In press), that allow them to visualize their programs in various ways.

- *Debugging*. To help find and diagnose bugs in their programs, programmers typically rely on some combination of print statements, hand simulation, and source-level debuggers (Eisenstadt, 1993). Alternatively, if they want to work with visual, as opposed to textual, representations of their programs, programmers may elect to enlist a so-called visual debugging system, such as the ParaDocs system used in Price's (1990) studies.

- *Performance tuning*. The advent of parallel computing has brought with it the need to identify and diagnose performance bottlenecks in parallel programs. A growing body of literature discusses the advantages of parallel performance visualization in that endeavor. Heath, Malony, & Rover (1995), for example, discuss several scenarios in which graphical views of parallel program performance can prove useful.

- *Software maintenance*. Industrial programs can grow to millions of lines. Keeping track of the evolution of such programs, and managing their further development, can prove a formidable task. In those endeavors, industrial programmers and managers might enlist visual summaries of various program attributes, such as those generated by SeeSys (Baker & Eick, 1995).

Table 14 provides a provisional assessment of the extent to which the SV effectiveness studies appear to be relevant to the leaf nodes of the taxonomy of Figure 16. (Those leaf nodes to which no studies appear relevant are omitted.) For each study or group of studies that are relevant to a leaf node, the table ranks the study or studies' ecological validity using a three-level rating system (*low*, *medium*, *high*). While the ratings are obviously subjective, the EXPLANATION/JUSTIFICATION column

[36]Interestingly, studies of programming teams in industry (e.g., Curtis & Walz, 1990) indicate that that software methods, tools, and environments play a secondary and limited role in productivity, . Other factors, such as the customer's involvement in requirements definition, project team experience, and team capability have been found to impact productivity more significantly than factors over which project management has control (e.g. software engineering practices and tools).

RECURRENT SOCIAL SCENE	RELEVANT STUDY	RATING	EXPLANATION/JUSTIFICATION
Lectures	(Badre *et al.*, 1991, §2.1)	☜	• Survey asks professors what resources they use to teach algorithms
	(Lawrence, 1993, ch. 9)	✌	• Lectures heard by students are much briefer than would be typical
	(Douglas *et al.*, 1995, 1996; Chaabouni, 1996)	✌	• Graduate students explain algorithms without an audience
Assignments	(Goldenson & Wang, 1991)	☜	• Study analyzes log files from actual student editing sessions • Researchers can only infer student intentions
	(Ford, 1993)	☟	• Study examines actual students working in teams on an "practical project" for the course in which they are enrolled • Ford's data collection method unclear, but he appears to use video-taping and interviews
	(Stasko, 1996)	☜	• Surveys of actual students in course provide empirical data • Stasko augments data with his own experiences in the course
Laboratories	(Price, 1990, Pilot and Experiment)	☜	• Finding a bug in a program someone else wrote seems plausible as a lab exercise, but not so plausible as an assignment
	(Hundhausen, 1993)	✌	• Students are asked to implement an algorithm and visualize it in Lens, or to visualize an algorithm already written for them • Students are not normally given a choice of algorithms to implement • Seems unlikely that students would receive a set of instructions for implementing a specific visualization
	(Wilson *et al.*, 1995)	☜	• Task seems typical of what students might do in a laboratory setting • Students work as a team, and the session lasts over an hour.
	(Byrne *et al.*, 1996, §2 and §3)	✌	• Students engage individually in structured prediction drills • Lab exercises are normally more involved than simply tracing through an algorithm • Unlikely that students would receive immediate feedback.
	(Kehoe & Stasko, 1996)	☜	• Students work at their own pace on a well-defined problem set
	(Mulholland, in press)	☜	• Students work in pairs on well-defined laboratory exercises
Study	(Badre *et al.*, 1991, §2.2; Stasko *et al.*, 1993; Lawrence, 1993, ch. 4 – 9; Reimer, 1996)	✌	• Study sessions with learning materials are far shorter (only 10 – 45 minutes) than would be typical in practice • Students are prohibited from studying in groups • Students are not allowed to study on their own time, or to use their preferred techniques
Tests	(Lawrence, 1993, ch. 8.3)	☜	• Students are shown animations and asked to write down rules describing fundamental properties and behavior
	(Byrne *et al.*, 1996, §2 and §3)	☜	• Showing students animations and having them predict the next snapshot seems a plausible means of using animations for tests
Team meetings	(Bellamy, 1993)	☟	• Study analyzes actual samples of programmers' notes; interviews are used for clarification
Software Development	(Bellamy, 1993)	☟	• Study analyzes actual samples of programmers' notes; interviews are used for clarification
Debugging	(Price, 1990, Pilot and Experiment)	☜	• Finding a bug in a 7,500 program someone else wrote (with a hint) seems reasonably pertinent to software debugging in industry

Table 14. Assessment of the ecological validity of the SV effectiveness studies. For each of the leaf level social scene included in the taxonomy of Figure 16, each study that appears relevant to that social scene is listed, along with its apparent degree of ecological validity (✌ = *low*, ☜ = *medium*, ☟ = *high*) of that study, and a short statement justifying its validity rating. Studies that seem relevant to multiple social scenes are included multiple times—once for each social scene to which they are relevant.

offers a rationale for each ranking. Some studies appear twice, indicating their possible relevance to two social scenes. On the other hand, three studies (Lawrence, 1993, ch. 4.2 and 4.3; Lavery & Cock-

ton, 1995) do not appear at all, indicating that they are too general to pertain directly to one of the leaf-node social scenes.[37]

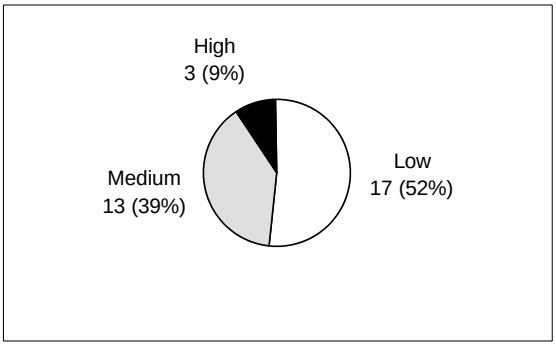

Figure 17. Summary of the ecological validity estimates offered by the assessment in Table 14.

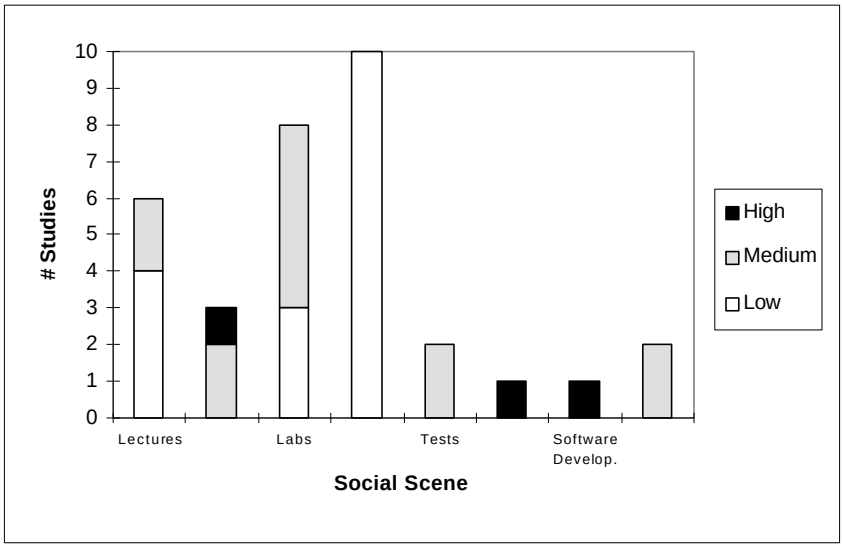

Figure 18. Summary of the ecological validity assessment of Table 14.

Figure 17 and Figure 18 further analyze the assessment made in Table 14, pointing out three noteworthy trends. First, a majority (52%) of the ecological validity assessments are *low*, while just under 10% are *high* (see Figure 17). With respect to this trend, it is important to keep two points in mind. First, three studies do not appear in the assessment, as I was unable to associate them with a social scene. Plainly, if I had chosen to associate those three studies with a social scene, their ratings would also have been *low*, increasing the overall percentage of *low* ecological validity ratings correspondingly. Second, five of the studies are classified twice, since they appear pertinent to two different social scenes. All five of these studies are rated *low* in one place, and *medium* in another. If I had imposed the restriction that each study could only be rated for one social scene, then clearly the percentage of low ratings would have done correspondingly, as I would have had to eliminate their five low ratings.

[37]Indeed, Lawrence's (1993, ch. 4.2 and 4.3) ranking studies do not suggest a setting in which such preferences are to be subsequently applied. Similarly, Lavery and Cockton's (1995) studies consider tasks that might be performed in just about any situation in which one would use the Napier software engineering workbench.

Second, as Figure 18 indicates, all ten of the studies that appear pertinent to the *study* social scene were judged to have *low* ecological validity. All of these studies have participants go through the same general procedure, in which they are first given a brief period of time to learn an algorithm with some learning medium (the *study* period), and in which they finish by completing a post-test to measure their understanding of the algorithm. Because of the obvious incongruities between this procedure and the way in which students actually study for tests in the real world, these studies earned a *low* rating.

Third, and in contrast, all four of the studies that address one of the *software engineering* social scenes garnered a *medium* or *high* rating. Two of these earn a high rating through their reliance on ethnographic field techniques to study industrial programmers (Bellamy, 1994), while the other two (Price, 1990, Pilot study and Experiment) earn a medium rating through their use of relatively large programs and tasks that appear reasonably pertinent to real-world debugging.

Finally, while a majority of studies (18 of 29) appear to be pertinent to the same two social scenes—*labs* and *study*—the ecological validity of those studies appears to be lower than the average across all studies (see Figure 18). Indeed, of the studies that are relevant to labs and study, 71% garner a *low* rating, with the remaining 29% receiving a *medium* rating. That a majority of the studies appear relevant to the same two social scenes probably reflects the overall trend, within the SV effectiveness studies, toward studying learning. In those studies that appear pertinent to *laboratories*, the learning sessions are highly-structured, with definite goals driving the learning. In those studies that appear pertinent to *study*, on the other hand, the learning is unstructured; students receive learning materials with no other instructions than to "study" for a test.

5. Synthesis: Summary and research agenda

While the notions of *usability* and *usefulness* may serve as reasonable preliminary characterizations of effectiveness, the preceding review and analysis suggest that the notion of SV effectiveness is far richer and more complex. Rather than having a meaning that is unambiguous and immutable (Gurka & Citrin, 1996), or that can be determined once-and-for-all outside of a context of use (Mackinlay, 1986), SV effectiveness, it appears, is actually an elusive concept whose definition is inextricably tied to the theory of effectiveness, research technique, and research foci of those performing the evaluation. In addition, I have illustrated that a chosen (a) theory of effectiveness, (b) research technique, and (c) research focus, heavily influence the degree to which an ensuing evaluation of effectiveness applies to SV in practice. In light of this meta-study's review and analysis of the SV effectiveness studies, which theories of effectiveness, research techniques, and research foci have been tried, which ones appear fruitful, and what avenues for future SV effectiveness research do they imply? I take up those questions in Sections 5.1, 5.2, and 5.3. Finally, in Section 5.4, I recap the degree to which the effectiveness studies apply to SV in practice, and I consider the implications of that applicability for future research.

5.1 Theories of effectiveness

ANALYSIS. An effectiveness study's theory of effectiveness rests on the study's division of labor, which both ascribes to SV a certain ontological status, and entrains epistemological assumptions about knowledge and its relation to SV. Studies that have examined the value of SV in learning about algorithms have evolved from theory in which SV is seen as an expert who facilitates efficient knowledge transfer (the *passive viewer* paradigm); to a theory in which SV is seen as a tool for self-guided problem exploration (the *active viewer* paradigms); to a theory in which SV is seen as a process in which one learns by becoming a teacher (the *visualization constructor* paradigm). The experiments conducted under the passive viewer paradigm have been markedly unsuccessful in substantiating effectiveness. In contrast, the empirical studies that have assumed an active viewer theory have yielded results that provide definite reason to be optimistic about SV's potential in learning.

Finally, while the *visualization constructor* paradigm remains largely unevaluated, the preliminary results (see Stasko, 1996) appear promising.

E **SYNTHESIS 1.** Researchers should abandon the passive viewer paradigm in favor of paradigms that view a visualization as a resource for engaging students in an active process of discovery, reflection, and explanation. While the two variations on the active viewer paradigm have been shown to be modestly effective in controlled experiments, researchers should strive to push the active envelope even further by devising new ways to engage students in active learning experiences. The visualization constructor paradigm suggested by Stasko (1996) appears promising, but more research is needed to understand and assess its potential benefits.

E **SYNTHESIS 2.** The SV effectiveness studies' intent focus on learning tasks has left more general theories of effectiveness largely unexplored. In the style of Table 10 (p. 26), Table 15 outlines two general theories of effectiveness that warrant further research. In accordance with Distributed Cognition Theory (see, e.g., Flor & Hutchins, 1991), Bellamy (1994) suggests that visualization serves as a form of *external memory*, reducing the cognitive complexity of the programming task. Alternatively, building on Situated Action Theory (see, e.g., Suchman, 1987), Roschelle (1990) argues that visualizations are one of a multitude of *mediational resources* that help groups of people to *negotiate* a shared understanding in both Visualization Exploration and Narration tasks.

PARADIGM	SUPPORTING STUDIES	ONTOLOGICAL STATUS OF SV	THEORY OF EFFECTIVENESS
External Knowledge	(Bellamy, 1994)	Externalizations of aspects of programming plans and solutions that do not lend themselves to expression in formal programming languages	SV reduces the cognitive complexity of the programming task by serving as external memory
Mediational Resource	(Roschelle, 1990)	Communicative resource akin to speech, gesture, and gaze	SV is a communicative resource for building a shared understanding of algorithms

Table 15. Two alternative theories of effectiveness that appear to be promising avenues for future research. The external knowledge paradigm addresses the task of Programming, while the Mediational Resource paradigm addresses various View Exploration and Narration tasks.

5.2 Research techniques

ANALYSIS. By far the most widely used evaluation technique, anecdotal evaluation lacks the empirical foundation that would make it plausible. Similarly, the scope of programmatic evaluation, another widely used technique, is too limited to make meaningful statements about effectiveness. By grounding their assessments in established design principles, specific SV systems, and real tasks, analytical techniques overcome many of the limitations of anecdotal and programmatic evaluation, proving useful in providing principled, rapid assessments of SV system usability. Despite these obvious advantages, their application is scant in the SV effectiveness literature.

E **SYNTHESIS.** While anecdotal and programmatic techniques may serve to capture initial interest in a SV technology development project, their lack of an empirical basis renders them ultimately unsatisfying. SV technology developers who are under pressure to disseminate their results rapidly should consider abandoning anecdotal and programmatic evaluation in favor of analytic techniques, which can be rapidly applied, and whose principled foundation may endow them with more persuasive power.

ANALYSIS. Drawing from six alternative empirical techniques, some 29 empirical studies of SV effectiveness have been published since the early 1990s. Most of the earliest studies were controlled experiments, with less rigorous observational and exploratory studies garnering increased interest as of late. Questionnaires and surveys have been widely used for both primary and complementary data, while usability tests and ethnographic field techniques have seen only limited use. As indi-

cated in Table 13 (p. 30), all of these empirical techniques hold promise in providing insight into various aspects of effectiveness; choosing an appropriate one is a largely a matter of finding a match to one's research questions and form of desired results. Those studies in which the research questions have focused on asserting causality have been less successful in obtaining their desired results (statistically-significant differences), whereas the surveys, questionnaires, and observational studies have been generally more successful in obtaining their desired results.

E **SYNTHESIS 1.** Despite the scientific allure of controlled experiments and the quantitative results they offer, past controlled experiments plainly point out the difficulty of asserting causality between learning medium and knowledge acquisition, even in tightly controlled environments. Aside from their potentially low ecological validity (see Section 4.4, p. 37),controlled experiments make a potentially "invalid implicit assumption" (Williams & Brown, 1990; see also Payne *et al.*, 1996) by "treat[ing] each medium as a more or less invariant entity with fixed clusters of attributes" (Williams & Brown, 1990, p. 219). Controlled experiments may also encounter difficulties in

- controlling for all of the significant variables (Gurka & Citrin, 1996),
- manipulating the correct variables (Kehoe & Stasko, 1996), and
- developing mesures that are sensitive to differences in learning promoted by alternative conditions (Kehoe & Stasko, 1996).

Researchers should thus think carefully before embarking on controlled experimental investigations of SV effectiveness; they may well enjoy more success with other techniques.

E **SYNTHESIS 2.** The six empirical techniques that past SV effectiveness studies have employed only begin to scratch the surface of the repertoire of approaches to studying computer-mediated activity being developed in the HCI community. Drawing from the social sciences, approaches such as Interaction Analysis (Jordan & Henderson, 1994), Distributed Cognition (see, e.g., Flor & Hutchins, 1991), and Activity Theory (see Nardi, 1996) develop increasingly sophisticated techniques for gathering and analyzing qualitative data. Future SV effectiveness research would do well to consider the potential for such techniques in developing and refining theories of effectiveness that have a strong empirical basis.

5.3 Research foci

ANALYSIS. The SV effectiveness research has examined collection of seven high-level research questions (see Figure 8, p. 32), which have been geared toward (a) determining the factors that influence learning with SV technology, (b) exploring the ways in which, and the reasons for which, humans make use of SV in various tasks, and (c) developing ways in which empirical data can be applied to SV technology design.

E **SYNTHESIS.** Missing from the collection of high-level research questions is any interest in the use of SV technology outside of academia. Future research should pose questions that allow it to move beyond educational settings. For example, can SV technology be used to increase productivity in industry, to which the research prototypes developed in academia are supposed to migrate? How might SV technology benefit the large software teams typical in industry? And what about the potential for SV technology to benefit the *distributed* programming teams of the future, who are linked only by the Internet?

ANALYSIS. Despite the fact that any scenario of SV use necessarily entrains Programming, Data and Event Selection, View Writing, and View Exploration, most of the SV effectiveness studies focus on View Exploration tasks (see Figure 13, p. 35); the other tasks are performed behind the scenes, precluding them from being grist for a broader analysis of entire scenarios of SV use.

E **SYNTHESIS.** While View Exploration is plainly the crux of any scenario of SV use, its prevalence in past studies has prevented them from obtaining a *holistic* perspective of effectiveness. Even if SV technology is found to be effective, in some sense, within the narrow scope of a View Exploration task, a whole host of other considerations could figure equally prominently in an overall assessment of effectiveness. For example, how long did it take to prepare and set up the SV technology and materials for the View Exploration task studied? Given the choice between using conventional materials and SV technology, which will instructors choose, and what considerations do they deem important in making that choice?

Beyond the scope of a particular scenario of SV use, researchers interested in evaluating the effectiveness of SV technology would also do well to consider the community within which the technology is to be deployed. As Bellamy (1996) notes,

> Teachers, administrators, parents, and others are also part of a learning situation. In order to effect change, systems of artifacts must be designed that address the needs of *all participants in the situation* and help them all move toward roles and ways of thinking appropriate for an alternative approach to education (p. 143, emphasis mine)

In sum, by assuming the narrow perspective of a single SV task, past SV effectiveness research has overlooked such important considerations. Accordingly, future research should broaden its scope to *entire scenarios of SV use*, and, ultimately, to *entire communities of practice.*

ANALYSIS. Most SV effectiveness studies have concentrated on scenarios in which undergraduate computer science students use XTango, POLKA, and a few other systems to learn about textbook algorithms through View Exploration tasks.

E **SYNTHESIS.** It is high time for SV effectiveness studies to broaden their repertoire of participants, target programs, and SV artifacts. While a few notable inroads have been made (e.g., Price, 1990; Bellamy, 1994), the literature generally lacks empirical studies of advanced or professional programmers engaged in SV tasks involving both large target programs, and SV systems with more than the standard play-and-pause interface.[38]

ANALYSIS. The potential ambiguity of open-ended learning tasks can lead to (a) experiments whose testing instruments do not cover the material that can be reasonably learned from the learning media, and (b) aimless discovery learning, which decreases the chances that participants will stumble upon the information and insight that could help them on the upcoming tests.

E **SYNTHESIS.** Future research should attempt to home in on SV's benefits by examining tasks that are more precisely defined. Indeed, the importance of firmly grounding the assumed benefits of visual representations in the details of *explicitly defined tasks* has been well demonstrated by Cassner and Larkin's (1989) empirical studies of relational graphics-reading tasks.

ANALYSIS. A secondary interest of the SV effectiveness research has been in studying novices and experts engaged in visualization programming (Data and Event Selection, View Writing) tasks, sometimes with pen, paper, and art supplies instead of with actual computer-based technology. These studies, along with at least one of the studies that investigated learning (Stasko et al., 1993), illustrate the importance of *narration* in situations in which SV plays an explanatory role.

[38]A clear parallel exists between the evolution of the literature on SV effectiveness and the literature on empirical studies of programmers (ESP) Indeed, owing to their origins in cognitive psychology, ESP researchers maintaned an intent focus on novice programmers during the first ten to fifteen years of the ESP literature's existence. It was not until the late 1980s and early 1990s, in the face of criticism that no one was studying real programmers (see Curtis, 1986), that ESP research began to investigate industrial programming practices. If the ESP literature's past evolution is any indication, we can expect the SV effectiveness studies to turn their attention to expert programmers and large programmers sometime shortly after the turn of the century!

- **SYNTHESIS.** Although past research into intelligent tutoring systems suggests that narration cannot be automated, it would appear that SV technology can better support the task of narration, which it has heretofore neglected. Future effectiveness research should thus concentrate on finding the appropriate union between SV technology and humans in the task of narration.

5.4 Ecological validity

ANALYSIS. SV does not amount to interpreting pictures in a vacuum; rather, SV occurs in the face of pressing practical concerns and against the backdrop of a larger enterprise, which itself has an objective. Those practical concerns, as well as the backdrop of the larger enterprise in which they arise, figure prominently in the ways in which SV is enlisted. A preliminary analysis of the relevance of the SV effectiveness studies to the social scenes in which SV use is widely held to occur (see Figure 16, p. 39) indicates that the ecological validity of a majority of the studies appears to be low, whereas the ecological validity of only 10% of the studies can be considered high.

- **SYNTHESIS.** That relatively few of the SV effectiveness studies appear to speak to the authentic practices of educators and those in industry should concern us. Future studies should make use of less tightly-controlled studies (e.g., those using ethnographic field techniques) before they use more tightly-controlled studies (e.g., controlled experiments, questionnaires), in order to ensure that they are investigating what actually counts.

ANALYSIS. A majority of the SV effectiveness studies appear relevant to the same two social scenes—study and laboratories; their ecological validity is lower than the average across all studies. On the other hand, few studies appear relevant to SV in industry; their ecological validity, however, is higher than the average across all studies.

- **SYNTHESIS.** In an attempt to produce empirical evidence in support of SV for learning, researchers appear to have sacrificed ecological validity, which would make their studies more readily applicable to practice. Future research should concentrate on firmly grounding the use of SV for learning in the real-world practices of professors and students through the use of ethnographic fieldwork. Although their ecological validity appears high, those few studies that appear relevant to industry only begin to address the role, present or potential, of SV technology in the actual practices of industrial programming teams. It is up to future research to build upon the empirical studies of programmers literature by providing detailed accounts that speak specifically to the advantages and disadvantages of SV technology within industrial settings.

Acknowledgments

I am grateful to my advisor, Sarah Douglas, for both challenging and inspiring me to go beyond a mere information dump of the SV literature. Had she not taken such interest in guiding me toward a more ambitious goal for this paper, I suspect that a much earlier (and far less sophisticated) version would have brought me just another pat on the back. In addition, Laura Girardeau merits special thanks for meticulously editing a draft of this article, and for providing unwavering support and encouragement through seemingly endless months of writing.

References

BADRE, A., BERANEK, M., MORRIS, J. M., & STASKO, J. T. (1991). Assessing program visualization systems as instructional aids. Technical report no. GIT-GVU-91-23. Graphics, Visualization, and Usability Center, Georgia Institute of Technology, Atlanta, GA.

BAECKER, R. (1981). *Sorting Out Sorting.* Color and sound film presented at SIGGRAPH '81. Toronto, Canada: Dynamic Graphics Project.

BAKER, M. J., & EICK, S. G. (1995). Space-filling software visualization. *Journal of Visual Languages and Computing 6*(2), 119-133.

BAZIK, J., TAMASSIA, R., REISS, S., & VAN DAM, A. (In press). Software visualization in teaching at Brown University. In M. Brown, J. Domingue, B. Price, & J. Stasko (Eds.), *Software visualization: Programming as a multimedia experience* . Cambridge, MA: The MIT Press.

BELLAMY, R. K. E. (1994). What does pseudo-code do? A psychological analysis of the use of pseudo-code by experienced programmers. *Human-Computer Interaction 9,* 225-246.

BELLAMY, R. K. E. (1996). Designing educational technology: Computer-mediated change. In B. Nardi (Ed.), *Context and Consciousness: Activity Theory and Human-Computer Interaction* (pp. 123-146). Cambridge, MA: The MIT Press.

BØDKER, S. (1996). Applying activity theory to video analysis: How to make sense of video data in HCI. In B. Nardi (Ed.), *Context and consciousness: Activity Theory and Human-Computer Interaction* (pp. 147-175). Cambridge, MA: The MIT Press.

BRAYSHAW, M., & EISENSTADT, M. (1991). A practical graphical tracer for Prolog. *International Journal of Man-Machine Studies 35*(5), 597-631.

BROWN, M.H., DOMINGUE, J., PRICE, B., & STASKO, J. (Eds.). (In press). *Software visualization: Programming as a multimedia experience* . Cambridge, MA: MIT Press.

BROWN, M. H. (1988*a*). *Algorithm animation*. Cambridge, MA: The MIT Press.

BROWN, M. H. (1988*b*). Perspectives on algorithm animation. In *Proceedings of the ACM SIGCHI '88 Converence on Human Factors in Computing Systems* (pp. 33-38). New York: ACM Press.

BROWN, M. H., & HERSHBERGER, J. (1991). Zeus: a system for algorithm animation. In *Proc. 1991 workshop on visual languages* (pp. 4-9). Los Alamitos, CA: IEEE CS Press.

BROWN, M. H., & SEDGEWICK, R. (1985). Techniques for algorithm animation. *IEEE Software 2*(1), 28-39.

BYRNE, M. D., CATRAMBONE, R., & STASKO, J. T. (1996). Do algorithm animations aid learning? Technical report no. GIT-GVU-96-18. Graphics, Visualization, and Usability Center, Georgia Institute of Technology, Atlanta, GA.

CASNER, S. M., & LARKIN, J. H. (1989). Cognitive efficiency considerations for good graphic design. In *Cognitive Science Society Proceedings* (pp. 275-282). Hillsdale, NJ: Erlbaum.

CHAABOUNI, Z. D. (1996). A user-centered design of a visualization language for sorting algorithms. Master's Thesis, Department of Computer and Information Science, University of Oregon, Eugene, OR.

CLEVELAND, W. S., & MCGILL, R. (1986). An experiment in graphical perception. *International Journal of Man-Machine Studies, 25,* 491-500.

COX, K. C., & ROMAN, G. C. (1994). An evaluation of the Pavane visualization system. Technical report No. WUCS-94-09. Department of Computer Science, Washington University in St. Louis, St. Louis, MO.

CURTIS, B. (1986). By the Way, Did Anyone Study Any Real Programmers? In *Empirical Studies of Programmers* (pp. 256-262). Norwood, NJ: Ablex.

CURTIS, B., & WALZ, D. (1990). the psychology of programming in the large: Team and organizational behavior. In J.-M. Hoc, T. R. G. Green, R. Samurcay, & D. J. Gilmore (Eds.), *Psychology of Programming* (pp. 253-270). San Diego: Academic Press.

DOUGLAS, S. A. (1995). Conversation analysis and human-computer interaction design. In P. Thomas (Ed.) In *Social and Interactional Dimensions of Human-Computer Interfaces* . Cambridge: Cambridge University Press.

DOUGLAS, S. A., HUNDHAUSEN, C. D., & MCKEOWN, D. (1995). Toward empirically-based software visualization languages. In *Proceedings of the 1995 IEEE Symposium on Visual Languages* (pp. 342-349). Los Alamitos, CA: IEEE Computer Society Press.

DOUGLAS, S. A., HUNDHAUSEN, C. D., & MCKEOWN, D. (1996). Exploring human visualization of computer algorithms. In *Proceedings 1996 Graphics Interface Conference* (pp. 9-16). Toronto, CA: Canadian Graphics Society.

EISENSTADT, M. (1993). Tales of debugging from the front lines. In *Empirical Studies of Programmers: Fifth Workshop* (pp. 86-112). Norwood, NJ: Ablex.

ERICSSON, K. A., & SIMON, H. A. (1984). *Protocol Analysis: Verbal Reports as Data*. Cambridge, MA: MIT Press.

FLOR, N. V., & HUTCHINS, E. L. (1991). Analyzing distributed cognition in software teams: A case study of team programming during perfective software maintenance. In J. Koenemann-Belliveau, T. G. Moher, & S. P. Robertson (Eds.), *Empirical Studies of Programmers: Fourth Workshop* (pp. 36-64). Norwood, NJ: Ablex.

FORD, L. (1993). How programmers visualize programs. Technical report No. R 271. Department of Computer Science, University of Exeter, Exeter, U.K.

GILMORE, D. J. (1990). Methodological issues in the study of programming. In J.-M. Hoc, T. R. G. Green, R. Samurcay, & D. J. Gilmore (Eds.), *Psychology of Programming* (pp. 83-98). San Diego: Academic Press.

GOLDENSON, D. R., & WANG, B. J. (1991). Use of Structure Editing Tools by Novice Programmers. In *Empirical Studies of Programmers: Fourth Workshop* (pp. 99-120). Norwood, NJ: Ablex.

GU, W. ET. AL.. (1994). Falcon: On-line monitoring and steering of large-scale parallel programs. Technical report no. GIT-CC-94-15. College of Computing, Georgia Institute of Technology, Atlanta, GA.

GURKA, J. S., & CITRIN, W. (1996). Testing effectiveness of algorithm animation. In *Proceedings of the 1996 IEEE Symposium on Visual Languages* (pp. 182-189). Los Alamitos, CA: IEEE Computer Society Press.

HEATH, M. T., & ETHERIDGE, J. A. (1991). Visualizing the performance of parallel programs. *IEEE Software, 8*(5), 29-39.

HEATH, M. T., MALONY, A. D., & ROVER, D. T. (1995). Parallel performance visualization: From practice to theory. *IEEE Parallel & Distributed Technology 3*(4), 44-60.

Hix, D., & HARTSON, H. R. (1993). Formative evaluation: Ensuring usability in user interfaces. In L. Bass & J. Dewan (Eds.) *User Interface Software* (pp. 1-30). New York: John Wiley & Sons.

HUNDHAUSEN, C. (1993). Exploring the potential for conversation analysis in the evaluation of interactive algorithm visualization systems. M.S. Project, Department of Computer and Information Science, University of Oregon, Eugene, OR.

HUNDHAUSEN, C. D. (1995). THESPIS: A framework for semantics-based software visualization interaction. Directed Research Project (available at http://www.cs.uoregon.edu/~chundhau/research). Department of Computer and Information Science, University of Oregon, Eugene, OR.

HUNDHAUSEN, C. D. (In preparation). Communicating about algorithms: From theory to design. Ph.D. Dissertation, Department of Computer and Information Science, University of Oregon, Eugene, OR.

JORDAN, B., & HENDERSON, A. (1995). Interaction analysis: Foundations and practice. *Journal of the Learning Sciences 4*(1), 39-103.

KEHOE, C. M., & STASKO, J. T. (1996). Using animations to learn about algorithms: An ethnographic case study. Technical report no. GIT-GVU-96-20. Graphics, Visualization, and Usability Center, Georgia Institute of Technology, Atlanta, GA.

KIMELMAN, D., ROSENBURG, B., & ROTH, T. (1994). Strata-Various: Multi-layer visualization of dynamics in software system behavior. In *Proceedings Visualization '94* (pp. 172-178). Los Alamitos, CA: IEEE Computer Society Press.

KRAEMER, E., & STASKO, J. T. (1993). The visualization of parallel systems: An overview. *Journal of Parallel and Distributed Computing 18*(2), 105-117.

LAVE, J. (1988). *Cognition in Practice*. Cambridge, MA: Cambridge University Press.

LAVERY, D., & COCKTON, G. (1995). A Pilot Study of Early Usability Evaluation Methods for Software Visualisations. FIDE Technical Report No. FIDE/95/141. University of Glasgow, Glasgow, Scotland.

LAWRENCE, A. W. (1993). Empirical studies of the value of algorithm animation in algorithm understanding. Ph.D. dissertation, Department of Computer Science, Georgia Institute of Technology, Atlanta.

LAWRENCE, A. W., BADRE, A. N., & STASKO, J. T. (1994). Empirically evaluating the use of animations to teach algorithms. In *Proceedings of the 1994 IEEE Symposium on Visual Languages* (pp. 48-54). Los Alamitos, CA: IEEE Computer Society Press.

MACKINLAY, J. (1986). Automating the Design of Graphical Presentations of Relational Information. *ACM Transactions on Graphics 5*(2), 110-141.

MAYER, R. E., & ANDERSON, R. B. (1992). Animations need narrations: An experimental test of a dual-coding hypothesis. *Journal of Educational Psychology 83*(4), 484-490.

MIYAKE, N. (1986). Constructive interaction and the iterative process of understanding. *Cognitive Science 10*, 151-177.

MUKHERJEA, S., & STASKO, J. T. (1994). Toward visual debugging: Integrating algorithm animation capabilities within a source-level debugger. *ACM Transactions on Computer-Human Interaction 1*(3), 215-244.

MULHOLLAND, P. (In press). A principled approach to the evaluation of SV: a case study in Prolog. In M. Brown, J. Domingue, B. Price, & J. Stasko (Eds.), *Software visualization: Programming as a multimedia experience* . Cambridge, MA: The MIT Press.

MYERS, B. A. (1986). Visual Programming, Programming by Example, and Program Visualization: A Taxonomy. In *Proceedings of ACM CHI'86 Conference on Human Factors in Computing Systems* (pp. 59-66). New York: ACM Press

MYERS, B. A. (1990). Taxonomies of visual programming and program visualization. *Journal of Visual Languages and Computing 1*(1), 97-123.

NAPS, T. (1990). Algorithm visualization in computer science laboratories. In *Proceedings of the 21st SIGCSE Technical Symposium on Computer Science Education* (pp. 105-110). New York: ACM Press.

NAPS, T. L. (Personal communication). E-mail correspondence dated January 30, 1997. .

NARDI, B. (Ed.). (1996). *Context and Consciousness: Activity Theory and Human-Computer Interaction* . Cambridge, MA: The MIT Press.

NIELSEN, J. (1992). Finding usability problems through heuristic evaluation. In *Proceedings of ACM CHI'92 Conference on Human Factors in Computing Systems* (pp. 373-380). New York: ACM Press.

PALMITER, S., & ELKERTON, J. (1993). Animated Demonstrations for Learning Procedural Computer-Based Tasks. *Human-Computer Interaction 8*(3), 193-216.

PANE, J. F., CORBETT, A. T., & JOHN, B. E. (1996). Assessing dynamics in computer-based instruction. In *Procedings of the 1996 SIGCHI Conference on Human Factors in Computing Systems* (pp. 197-204). New York: ACM Press.

PAPERT, S. (1980). *Mindstorms: Children, Computers and Powerful Ideas*. New York: Basic Books.

PENNINGTON, N., & GRABOWSKI, B. (1990). The tasks of programming. In J.-M. Hoc, T. R. G. Green, R. Samurcay, & D. J. Gilmore (Eds.), *Psychology of Programming* (pp. 45-62). San Diego: Academic Press.

PETRE, M. (1995). Why looking isn't always seeing: Readership skills and graphical programming. *Communications of the ACM 38*(6), 33-44.

PETRE, M., & GREEN, T. R. G. (1993). Learning to read graphics: Some evidence that 'seeing' an information display is an acquired skill. *Journal of Visual Languages and Computing 4*, 55-70.

POLSON, P. G., LEWIS, C., RIEMAN, J., & WHARTON, C. (1992). Cognitive Walkthroughs: A Method for Theory-Based Evaluation of User Interfaces. *International Journal of Man-Machine Studies 36*(5), 741-773.

PRICE, B. (1990). *A framework for the automatic animation of concurrent programs*. M.S. Thesis, University of Toronto.

PRICE, B. A., BAECKER, R. M., & SMALL, I. S. (1993). A principled taxonomy of software visualization. *Journal of Visual Languages and Computing 4*(3), 211-266.

REDDY, M. (1979). The conduit metaphor: A case of frame conflict in language about language. In A. Ortony (Ed.), *Metaphor and Thought* (pp. 66-102). Cambridge: Cambridge University Press.

REIMER, Y. (1996). Task-Based Design of Algorithm Animations. Master's Thesis, Department of Computer Science, University of Montana, Missoula, MT.

REISS, S. (In press). Visualization for software engineering--Programming environments. In M. Brown, J. Domingue, B. Price, & J. Stasko (Eds.), *Software visualization: Programming as a multimedia experience* . Cambridge, MA: The MIT Press.

RIEBER, L. P. (1990). The effects of animated graphics on student learning. *Journal of Educational Psychology 82*(??), 123-???

RIEBER, L. P., BOYCE, M. J., & ASSAD, C. (1990). The effects of computer animation on adult learning and retrieval tasks. *Journal of Computer-Based Instruction 17*(2), 46-52.

ROMAN, G. C., & COX, K. C. (1993). A taxonomy of program visualization systems. *IEEE Computer 26*(12), 11-24.

ROMAN, G. C., COX, K. C., WILCOX, C. D., & PLUN, J. Y. (1992). Pavane: A system for declarative visualization of concurrent computations. *Journal of visual languages and computing 3*(2), 161-193.

ROSCHELLE, J. (1990). *Designing for conversations*. Paper presented at the AAAI Symposium on Knowledge-Based Environments for Learning and Teaching, Stanford, CA.

SANDERSON, P. M., & FISHER, C. (1994). Exploratory sequential data analysis: Foundations. *Human-Computer Interaction 9*(3-4), 251-318.

SANJEK, R. (1995). Ethnography. In A. Barnard & J. Spencer (Eds.) In *Encyclopedic dictionary of social and cultural anthropology* . London: Raitledge.

SHU, N. C. (1988). *Visual programming*. New York: Van Nostrand Reinhold.

SINGH, G., & CHIGNELL, M. H. (1992). Components of the visual computer: A review of relevant technologies. *Visual Computer 9*, 115-142.

SPRADLEY, J. P. (1979). *The Ethnographic Interview*. New York: Holt, Rinehart, and Winston.

SPRADLEY, J. P., & MCCURDY, D. W. (1972). *The Cultural Experience: Ethnography in Complex Society*. Palo Alto, CA: Science Research Associates.

STASKO, J. (1989). Tango: A framework and system for algorithm animation. Ph.D. dissertation, Computer science, Brown University, Providence, RI.

STASKO, J. T. (1992). Animating algorithms with XTango. *SIGACT News, 23*(2), 67-71.

STASKO, J. T. (1996). Using student-built algorithm animations as learning aids. Technical report no. GIT-GVU-96-19. Graphics, Visualization, and Usability Center, Georgia Institute of Technology, Atlanta, GA.

STASKO, J., BADRE, A., & LEWIS, C. (1993). Do Algorithm Animations Assist Learning? An Empirical Study and Analysis. In *Proceedings of ACM INTERCHI'93 Conference on Human Factors in Computing Systems* (pp. 61-66).

STASKO, J., & KRAEMER, E. (1993). A methodology for building application-specific visualizations of parallel programs. *Journal of parallel and distributed computing, 18*(2), 258–264.

STASKO, J. T., & MUTHUKUMARASAMY, J. (1996). Visualizing program executions on large data sets. In *Proceedings of the IEEE Symposium on Visual Languages* (pp. 166-173). Los Alamitos, CA: IEEE Computer Society Press.

STASKO, J. T., & PATTERSON, C. (1992). Understanding and characterizing software visualization systems. In *Proceedings of the 1992 IEEE Symposium on Visual Languages* (pp. 3-10). Los Alamitos, CA: IEEE Computer Society Press.

SUCHMAN, L. A. (1987). *Plans and Situated Actions: The Problem of Human-Computer Communication*. New York: Cambridge University Press.

TUFTE, E. R. (1983). *The Visual Display of Quantitative Information*. Cheshire, CT: Graphics Press.

WILLIAMS, C. J., & BROWN, W. W. (1990). A review of research issues in the use of computer-related technologies for instruction: What do we know? *Journal of Instructional Media 17*(3), 213-225.

WILSON, J., KATZ, I. R., INGARGIOLA, G., AIKEN, R., & HOSKIN, N. (1995). Students' use of animations for algorithm understanding. In *CHI '95 Proceedings* (pp. 238-239). New York: ACM Press.

WOLCOTT, H. F. (1992). Posturing in qualitative inquiry. In M. D. LeCompte, W. L. Millroy, & J. Preissle (Eds.),*The Handbook of Qualitative Research in Education* (pp. 3-52). San Diego: Academic Press.

Appendix: Summary of the meta-study data

The following table provides a synopsis of the raw data upon which the analysis of Section 4 is based. For each SV effectiveness study (column 1), the table lists the research technique employed (column 2), research questions examined (column 3), the target programs (column 4) and SV technology (column 5) considered, the kinds of people who participated in the study (column 6), the tasks they performed (column 7) considered, and the assessment of ecological validity offered in Section 4.4 (column 8). The abbreviations for SV tasks used in column 7 are the same as those introduced in Section 4.1.1 (p. 23), and the symbols used to indicate ecological validity are the same as those introduced in Section 4.4. All other abbreviations, including those used as subscripts in columns 7 and 8, are explained in the tables of abbreviations on the following page.

STUDY	RESEARCH TECHNIQUE	RESEARCH QUESTIONS	PROGRAM(S)	SV TECHNOLOGY	PARTIC-IPANTS	TASKS	ECO. VALIDITY
(Price, 1991, Exp.)	CE, OS, EFT	κ	7,500 line OS simulation	Paradocs	G	VE_2	☜$_{La, D}$
(Price, 1991, Pilot)	UT	ν	7,500 line OS simulation	Paradocs	G	VE_2	☜$_{La, D}$
Stasko *et al.*, 1993	CE, QS	φ	Pairing heap implementation of priority queue	XTango	UG	VE_1, IDS	✌$_S$
(Lawrence, 1993, ch. 4.2)	QS	ν	Quick sort	XTango	UG	VW	—
(Lawrence, 1993, ch. 4.3	QS	ν	Sorting algorithms	XTango stills	UG	VW	—
(Lawrence, 1993, ch. 4.4	CE	φ	Quick sort	XTango	UG	VE_1	✌$_S$
(Lawrence, 1993, ch. 5)	CE	φ	Quick sort, selection sort	XTango	UG	VE_1	✌$_S$
(Lawrence, 1993, ch. 6)	CE	φ	Kruskal's MST, quick sort	XTango	UG	VE_1, IDS	✌$_S$
(Lawrence, 1993, ch. 7)	CE	φ	Kruskal's MST	XTango	UG	VE_1, IDS	✌$_S$
(Lawrence, 1993, ch. 8.2)	CE	φ	Selection sort, Kruskal's MST	XTango	UG	VE_1, IDS	✌$_S$
(Lawrence, 1993, ch. 8.3)	OS	φ	Selection sort, radix sort quick sort	XTango	UG	VE_1	✌$_S$ ☜$_T$
(Lawrence, 1993, ch. 9)	CE	φ	Kruskal's MST	XTango, POLKA, POLKA stills	UG	VE_1, IDS	✌$_S$ ✌$_{Le}$
(Byrne *et al.*, 1996, §2)	CE	φ	Depth-first search	POLKA	UG	VE_1	✌$_{La}$ ☜$_T$
(Byrne *et al.*, 1996, §3)	CE	φ	Binomial heap implementation of priority queue	POLKA	UG	VE_1	✌$_{La}$ ☜$_T$
(Mulholland, In press)	CE, OS	κ	Short Prolog program	TPM	UG	VE_3	☜$_{La}$
(Badre *et al.*, 1991, §2.1)	QS	λ, θ	Shellsort	Pen and paper	P	IDS, VW	☜$_{Le}$
(Badre *et al.*, 1991, §2.2	OS, EFT	ο	Shellsort	XTango	UG	VE_1	✌$_S$
(Goldenson & Wang, 1991)	OS	μ	Student assignments	Pascal Genie	UG	P, VE_4	☜$_A$
(Ford, 1993)	OS, EFT	λ	Imperative programming con-structs	Pen, Paper, Goofy	UG	IDS, VW	👎$_A$
(Wilson *et al.*, 1995)	OS	μ	A*, depth-first search	FLAIR	UG	VE_3	☜$_{La}$
(Douglas *et al.*, 1995)	OS	λ ν	Bubble sort	Art supplies, Lens	G	DES, VW, VE_5, N	✌$_{Le}$
(Douglas *et al.*, 1996)	OS	ν λ	Bubble sort	Art supplies, Lens	G	DES, VW, VE_5, N	✌$_{Le}$
(Chaabouni, 1996)	OS	ν λ	Insertion sort, heap sort, quick sort	Art supplies	G	DES, VW, N	✌$_{Le}$
(Kehoe & Stasko, 1996)	OS	μ	Binomial heap implementation of priority queue	Web browser w/POLKA anims and stills	G	VE_6	☜$_{Le}$
(Reimer, 1996)	OS, QS	ν	Graham scan convex hull	Custom Hypercard Stack	UG	VE_1	✌$_S$
(Stasko, 1996)	QS	φ	Quick sort, MST	Samba	UG	DES, VW, IDS, VE_4	☜$_A$
(Bellamy, 1994)	EFT	π	Various authentic projects	Pen, paper	PP	P, DES, VW, IDS, VE_7	👎$_{TM}$ 👎$_{SD}$
(Hundhausen, 1993)	UT	ν	bubble sort, insertion sort, self-designed algorithms	Lens	G	P, DES, VW, IDS, $VE_{2,5}$	✌$_{La}$
(Lavery & Cockton, 1995)	AT, UT	ν	S.E. environment	Custom SV tools for Napier S.E. environment	UG	VE_8	—

Table A-1. Summary of raw data analyzed in Section 4 of meta-study

Appendix (cont.): Explanation of abbreviations in Table A-1

ABBREVIATION	RESEARCH TECHNIQUE
CE	Controlled Experiment
OS	Observational Study
QS	Questionnaire/Survey
EFT	Ethnographic Field Technique
UT	Usability Test
AT	Analytical Technique

Table A-2. Two- or three-letter codes used to abbreviate the research techniques in column 2

ABBREVIATION	GENERAL RESEARCH QUESTION
φ	What factors influence learning with SV?
κ	How do SV tracers influence problem solving?
λ	How do humans visualize algorithms and programming constructs?
μ	How is SV used in various View Exploration tasks?
ν	How can empirical data improve the design of SV?
ο	How can we evaluate algorithm understanding?
π	How do programmers use notation in programming?
θ	How do instructors teach algorithms?

Table A-3. Circled numbers used to abbreviate the general research questions in column 3

ABBREVIATION	PARTICIPANT TYPE
UG	Undergraduate Student
G	Graduate Student
P	Professor
PP	Professional Programmer

Table A-4. One- or two-letter codes used to abbreviate the participant types in column 6

ABBREVIATION (VE SUBSCRIPT)	VE TASK
1	Learn algorithm (prepare for a test)
2	Find bug in program
3	Figure out how or where two programs differ
4	Complete programming assignment
5	View the visualization you just programmed
6	Answer set of questions on an algorithm
7	Provide samples of the pseudocode you've created in your day-to-day programming activities
8	Common search tasks in Napier S.E. environment

Table A-5. Subscripts used to describe more precisely the VE tasks in column 7.

ABBREVIATION (ECO. VALIDITY SUBSCRIPT)	RECURRENT SOCIAL SCENE TO WHICH ECOLOGICAL VALIDITY RATING APPLIES
Le	Lectures
S	Study
A	Assignments
La	Laboratories
T	Tests
TM	Team meetings
SD	Software Development
D	Debugging

Table A-6. Subscripts used to indicate recurrent social scene to which ecological validity ratings in column 8 are relevant