It has been a long way since Knowlton’s movie about list processing with the programming language L6 [14]. Thousands of algorithm animations, hundreds of systems, and numerous case studies and evaluations have been produced since. But don’t get me wrong, it’s not all said and done? By and large software visualization research has concentrated on a few aspects of software. So you might ask, what should it concentrate on in the future? To answer this question we present a quantitative map of existing research and discuss some cross-topic research themes.

There exist several taxonomies for software visualization in the literature [20, 22, 23, 25]. We propose another one here for the sole purpose of identifying almost unexplored research areas. The taxonomy is based on what and not why or how it is visualized.

Certainly there are various dimensions which could be used to categorize research in software visualization. In Figure 1 we use two dimensions: the classical abstraction layers of a software system (hardware, virtual/abstract machine, program and system) and the static and dynamic phenomena of these layers. The map is incomplete in the sense that one could add additional layers (e.g. operating system) or structures (e.g. project structure). In addition to the qualitative information we use shades of gray to indicate how much published research exists in certain areas of software visualization. Dark gray indicates a high number (more than 100), medium gray a low number (more than 10) and light grey almost no published research. The numbers are based on web searches using two comprehensive search engines, namely CiteSeer (www.citeseer.com) and Google (www.google.com). For the search we used the abstraction layers as keywords.

---

1 Our taxonomy is closest to Myers’ taxonomy [20]. He proposes a 2 × 3 matrix (\{dynamic, static\} × \{code, data, algorithm\}), but provides no quantitative analysis.
together with “visualization” or “software visualization”. E.g. for “algorithm visualization”, CiteSeer found 22 papers with these keywords in their titles, for “program visualization” it found more than 300. Then we looked at the titles, abstracts or contents to decide which aspect is covered by a paper. For the combination “abstract machine” and “visualization”, CiteSeer found no entries. Using Google and browsing through 845 web pages which contained these keywords, we found that there are more than 10 papers which address the topic of visualizing abstract machines. As the same concepts go by different names in different communities this study does not claim to be complete or comprehensive in any way, but we think that it gives a rough orientation on the activity of research in these areas.

1 Discussion and Examples

We will now discuss some of the regions in this research map and give examples of prototypical or seminal work in the more unexplored regions including references to articles in this volume.

Static Structure. The static structure of programs and systems has been visualized in various ways including pretty printing, control-flow diagrams, and UML diagrams. The reader certainly will have seen many of those before. Korhonen et. al. [15] suggest to have students run a program once or more with different input data and indicate the path coverage of those input data in the control flow diagram.

Concrete Execution. The execution of (small) programs has been visualized in various ways and many animations are accessible on the internet. The blurry distinction of program and algorithm animations based on the level of abstraction used in the visualization is widely accepted. Visualization systems to automate the production of such animations have been created. They mostly differ in what kinds of data structures and algorithms are visualized, and what programming languages are used. Program animations have been mainly used in education.