The Evolving User-Centered Design of the Algorithm Visualization Storyboarder

Christopher Hundhausen
Visualization and End User Programming Lab
College of Electrical Eng. and Computer Science
Washington State University
Pullman, WA 99164-1752
hundhaus@eecs.wsu.edu

Joshua Wingstrom, Ravikiran Vatrapu
Laboratory for Interactive Learning Technologies
Information and Computer Sciences Department
University of Hawai`i
Honolulu, HI 96822
{wingstro, ravikira}@hawaii.edu

Abstract

The ALgorithm VIsualization Storyboarder (ALVIS) enables beginning and intermediate computer science students both to construct "low fidelity" (rough, unpolished) visualizations of the algorithms under study, and to present those visualizations to their peers and instructor for feedback and discussion. The original ALVIS software [3] was a frail research prototype. We are using an iterative, user-centered design process to develop a public domain version of the software suitable for use in a "studio-based" computer science course. Through a series of empirical studies of both low and high fidelity prototypes of ALVIS, we have reevaluated the original design of ALVIS, and are presently in the process of making three key design changes: (a) the underlying scripting language is being redesigned for simplicity and ease of learning; (b) the interface for creating and placing graphical variables and spatial structures is being simplified and streamlined for ease of use; and (c) new features are being added that explicitly support storytelling.

1. Introduction

Firmly rooted in the real-world activity of constructing a "visualization storyboard" [1] out of simple art supplies, the ALgorithm VIsualization Storyboarder (ALVIS) enables beginning and intermediate computer science students both to construct "low fidelity" (rough, unpolished) visualizations of the algorithms under study, and to present those visualizations to their peers and instructor for feedback and discussion.

The original ALVIS software [4] was a frail research prototype created solely to explore the design space of "low fidelity" algorithm visualization. In ongoing work, we are developing a public domain version of the software suitable for use in a "studio-based" introductory or intermediate computer science course that revolves around the construction and discussion of "low fidelity" algorithm visualizations [3].

In addition to being more robust, our public domain version of ALVIS differs from the original version in three significant ways:

1. the underlying scripting language (SALSA) is being redesigned for simplicity and ease of learning;
2. the interface for creating and placing graphical variables and spatial structures is being simplified and streamlined for ease of use; and
3. new features are being added that explicitly support storytelling, an aspect of the "studio-based" approach that we believe to have important educational value.

In this technical note, we elaborate on these changes, and ground the changes in the empirical data we have gathered through our iterative, user-centered design process.

2. Redesign of ALVIS

Over the past year, 18 second-semester computer science students have participated in empirical studies involving both high and low fidelity prototypes of ALVIS. Data collected in these studies have suggested the need for three key design changes, as detailed in the following subsections.

2.1 Changes to Underlying SALSA Language

ALVIS is a front-end interactive environment for programming in SALSA (Spatial Algorithmic Language for StoryboArding), an interpreted scripting language for specifying algorithm visualizations. We are in the process of making three fundamental changes to the SALSA language based on our empirical study results.

First, we observed in our studies that SALSA users spent inordinate amounts of time generating data values for their algorithms—a process in which they had to perform numerous drag-and-drop, dialog box fill-in, and text editing operations. To address this issue, we are presently incorporating into SALSA explicit support for automatic random data generation. Under the new design, one can specify the value of a variable to be "random" when it is created:

```
create variable foo -value random integer
  between 1 and 100
```

Then, when the variable is cloned, e.g.,

```
create variable bar as clone of foo
```

a new random data value is generated for the cloned variable. Through an "auto populate" feature being explored for spatial structures such as arrays and trees (see next section), a user can quickly populate a spatial structure with random data.

Second, past descriptions of SALSA (e.g., [4]) have emphasized its explicit support for spatial relations as a means of modeling algorithmic logic. Based on our usability study results, we discovered that general spatial relations are problematic because their interpretation depends heavily on context. For example, consider a loop in which we walk a variable `arrow` through all five cells of an array `myarray`. Here is one possible loop construction in the old SALSA syntax:

```
while arrow is-left-of right-center of
       variable at cell 5 of myarray
  --do something
  move arrow right cellwidth of myarray
endwhile
```

In this case, the `is-left-of` relation means "generally left of." That is, as long as `arrow` remains left of the right edge of the last element in `myarray`, the relation holds. In contrast, in some situations, programmers would like to formulate loop logic in terms of "direct" spatial relations semantics. Consider, for example, the following alternative formulation of the same loop logic expressed in the previous example:

```
while variable is-below arrow
  --do something
  move arrow right cellwidth of myarray
endwhile
```

In this case, the `is-below` relation means "strictly below." In other words, as long as there exists a variable that is directly below `arrow`, the loop should proceed. When `arrow` advances to whitespace past the final array element, the loop is done.

The extent to which spatial relations are contextually dependent has prompted us to eliminate spatial relations-driven branching and looping in SALSA. The new version of SALSA supports only the spatial relations semantics explicitly defined by spatial structures—for example, "left-of" and "right-of" relations between variables within an array.

Third, even though study participants were asked to program basic array iteration algorithms with which they were already familiar, they had difficulties formulating the correct iterative logic for these algorithms in SALSA. Given that prior empirical studies have shown that loops are notoriously difficult for beginning programmers to write [6], we realized that the SALSA language needs to provide better support for writing loops. To that end, we have replaced the confusing spatial-relations-driven loop constructs illustrated in the previous paragraphs with an "iterator" data type. To iterate over an array, the user can associate an iterator with the array, and then use the iterator to walk through the array elements. The following code segment uses the new SALSA `arrayiterator` data type to formulate the same loop logic expressed in the previous two examples:

```
create arrayiterator iter over myarray
  -marker arrow --use arrow var. as a marker
while iter is-valid
  --do something
  move iter right
endwhile
```

Note that `is-valid` is a special operator that returns true if an iterator is positioned at a valid array location, false otherwise.

2.2 Redesign of Variable Creation Interface

To create and place a variable in the original ALVIS interface, users not only had to fill in a series of dialog boxes containing variable attribute settings, but also had to explicitly drag and drop a newly-created variable into the animation view. In empirical studies of the original ALVIS prototype, participants had difficulties negotiating these dialog boxes. Moreover, we found that participants expected to be able to place variables in the animation view at the same time they created them, instead of having to explicitly place them after creating them.

These problems stemmed not only from the complexity of the SALSA language, which defines numerous attributes for variables (e.g., "graphical representation," outlinecolor") and separate "create" and "place" commands, but also from the nested sequence of dialog boxes and drag-and-drop actions that had to be negotiated in order to create a new variable. In interviews with study participants, we learned that they did not care to know about all of the possible attribute settings at variable creation time; they were more interested in quickly placing a variable into the animation view.

In a follow-up user-centered design project inspired by the above observations, we are in the process of completely redesigning the variable creation interface. Using "minimalist" design principles, we have simplified the variable creation dialog box by eliminating four variable attributes that we deemed were unnecessary. In addition, in response to our earlier usability study results, we have combined variable creation and placement into a single SALSA "create" command, as illustrated by the following example:

```
create variable foo at cell 1 of myarray
```

To streamline the generation of variable creation commands in the ALVIS environment, we have created a

"variable" tool. With a single click, the "variable" tool enables a user to create a variable with default attributes and place it in the animation view at the location clicked. If the default attributes are not to the user's liking, the user can double-click on the newly-created variable to obtain a dialog box through which the variable's attributes can be changed as needed.

Note that this general "minimalist" design approach is also guiding the design of ALVIS tools that support the creation, placement, and automatic population of spatial structures such as arrays and trees. For example, using a new "array" tool, the user can drag out a bounding box for an array at a specific location in the animation view; the number of rows and columns of the array is dynamically adjusted in response to mouse movements. The user can then populate the array with data by selecting the "autopopulate wand" and clicking anywhere within the array. A pop-up menu then appears, from which the user can select the variable to be used as a template for cloning the variables with which to populate the array.

2.3 Explicit Support for Storytelling

In our ethnographic studies of visualization assignments in an undergraduate, third-year algorithms course [2], we observed that students took great pleasure in creating visualizations that portrayed a given algorithm in terms of a story, and that visualizations with story content tended to stimulate increased audience interest and involvement. While the original version of ALVIS ignored this observation, we believe that the interest and involvement stimulated by visualizations with story content are key benefits of the "studio-based" teaching approach supported by ALVIS [3]. Accordingly, we are presently redesigning ALVIS so that it explicitly supports storytelling. Storytelling features presently being explored through our user-centered design process include the following:

- the ability to choose a background canvas for the visualization (the "setting" of the story") from a library of themes, or to sketch out a new one;
- the ability to add dialog to animations through "speaking bubbles" that appear above animation objects as they speak, as in end-user programming systems like ALICE [5]; and
- the ability to explore possible storylines for a visualization through the use of an optional storytelling wizard that asks a series of strategic questions designed to provide guidance and inspiration.

3. Summary and Current Status

In this technical note, we have described the evolution of the ALVIS algorithm visualization system in terms of three key design changes that were motivated by a series of empirical studies with low and high fidelity prototypes. In so doing, we have not only demonstrated the value of applying an iterative, user-centered design process to the development of end user programming environments, but we have also illustrated the translation of our empirical results into concrete design features that support ease of use, learnability, and storytelling.

We anticipate that the design changes described above will continue to evolve through iterative refinement and user testing. A preliminary release version of the ALVIS software that includes these changes should be available in the public domain by late summer of 2004; it will be used in a pilot offering of a studio-based introductory computer science course in the fall of 2004.

For updates on our progress, and for information on how to obtain ALVIS, please visit the "Algorithms Studio" project website at

http://eecs.wsu.edu/~veupl/proj/algstudio

4. Acknowledgments

Hank Bennett and Chad Takahashi helped to implement the original ALVIS prototype. Presently, Jonathan Lee Brown and Sean Farley are active participants in the design and implementation of ALVIS. This work was supported by the National Science Foundation under grant no. 0133212.

5. References

[1] S. A. Douglas, C. D. Hundhausen, and D. McKeown, "Toward empirically-based software visualization languages," in *Proceedings of the 11th IEEE Symposium on Visual Languages*. Los Alamitos, CA: IEEE Computer Society Press, 1995, pp. 342-349.

[2] C.D. Hundhausen, Integrating algorithm visualization technology into an undergraduate algorithms course: Ethnographic studies of a social constructivist approach. *Computers & Education*, vol. 39 pp.. 237-260.

[3] C.D. Hundhausen, "The "Algorithms Studio" Project," in *Proc. IEEE HCC '02*. Los Alamitos, CA: IEEE Computer Society Press, 2002, pp. 99-100.

[4] C. D. Hundhausen and S. A. Douglas, "Low fidelity algorithm visualization," *J. Visual Languages and Computing*, vol. 13, pp. 449-470, 2002.

[5] R. Pausch, T. Burnette, A. C. C. M. Conway, D. Cosgrove, R. DeLine, J. Durbin, R. Gossweiler, S. Koga, and J.White, "Alice: Rapid Prototyping for Virtual Reality," *IEEE Computer Graphics and Applications*, vol. 15, pp. 8-11, 1995.

[6] E. Soloway, J. Bonar, and K. Ehrlich, "Cognitive strategies and looping consructs: an empirical study," *Communications of the ACM*, vol. 26, pp. 853-860, 1983.