BRIDGES: A System to Enable Creation of Engaging Data Structures Assignments with Real-World Data and Visualizations

David Burlinson, Mihai Mehedint, Chris Grafer, Kalpathi Subramanian, Jamie Payton, Paula Goolkasian
University of North Carolina at Charlotte
{dburlins, mmehedin, cgrafer1, krs, payton, pagoolka}@uncc.edu

Michael Youngblood
PARC, A Xerox Company
Michael.Youngblood@parc.com

Robert Kosara
Tableau Research
rkosara@tableau.com

ABSTRACT

Although undergraduate enrollment in Computer Science has remained strong and seen substantial increases in the past decade, retention of majors remains a significant concern, particularly for students at the freshman and sophomore level that are tackling foundational courses on algorithms and data structures. In this work, we present BRIDGES, a software infrastructure designed to enable the creation of more engaging assignments in introductory data structures courses by providing students with a simplified API that allows them to populate their own data structure implementations with live, real-world, and interesting data sets, such as those from popular social networks (e.g., Twitter, Facebook). BRIDGES also provides the ability for students to create and explore *visualizations* of the execution of the data structures that they construct in their course assignments, which can promote better understanding of the data structure and its underlying algorithms; these visualizations can be easily shared via a weblink with peers, family, and instructional staff. In this paper, we present the BRIDGES system, its design, architecture and its use in our data structures course over two semesters.

Keywords

algorithm, data structure, visualization, engagement

1. INTRODUCTION

Over the past several years, enrollment in computer science undergraduate degree programs has been increasing at rates ranging from 6-11% [9]. Despite this increase, retention of CS majors remains a critical concern, especially at the freshmen/sophomore levels, with attrition rates of up to 66% [8]. While the factors that motivate changing majors are many and varied [8], strategies to increase retention are crucial to meet the national needs for graduates that can contribute to a robust and innovative twenty-first century workforce.

Demonstrating a connection between computing and the real world has the potential to increase students' motivation and interest in computing; grounding teaching in familiar, concrete, and relevant examples has been shown to improve learning [4] and has a positive effect on the retention of computing majors, particularly for women students [7]. In practice, however, there is little support for educators who want to incorporate this kind of approach into introductory computer science courses; for most students in introductory computer science courses, there remains a disconnect between the students' own everyday experiences with existing information systems that address real-world problems and the toy examples with small, uninteresting, synthetic data sets they encounter in course assignments that are used to teach students how data structures and algorithms work.

In this paper, we introduce the **BRIDGES** (**B**ridging **R**eal-world **I**nfrastructure **D**esigned to **G**oal-align, **E**ngage, and **S**timulate) system, which (1) facilitates student access to live, real-world data sets for use in traditional data structures programming assignments and (2) makes it possible for students to view and verify (debug) *their own implementations* of data structures, by providing *visualization* capabilities. The goal is to increase the engagement of students enrolled in data structures courses in an effort to improve retention of students in the computer science major. We present the architecture of BRIDGES and sample projects that can be enabled through the use of BRIDGES in a data structures and algorithms course. We describe a study of the application of BRIDGES in two semesters of a sophomore-level course; results show that the majority of students indicate that the BRIDGES-enabled assignments are relevant to career goals and increased their interest in computing.

SIGCSE '16, March 02-05, 2016, Memphis, TN, USA

DOI: http://dx.doi.org/10.1145/2839509.2844635

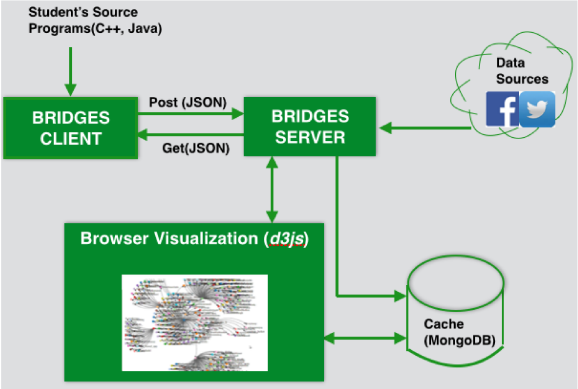


Figure 1: BRIDGES System. Based on a client-server architecture; a user constructs data structure programs using the BRIDGES client classes (Fig. 3) in Java or C++. The data structure representation is sent to the server by the client as a JSON string. After parsing and user authentication, the server stores the assignment on the MongoDB and displays the data structure on a specific web link using D3. The server also has interfaces to retrieve data from external sources upon user requests for use in their programs.

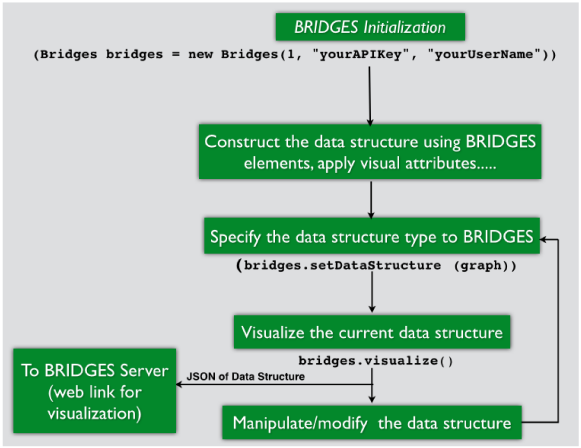


Figure 2: An Example BRIDGES program. Consists of an initialization step to specify user name and id (for authentication), assignment number. This is followed by user's construction of the data structure using BRIDGES client classes. The final 2 steps involve specifying a handle to the data structure (tree root, graph, head of a linked list, etc) and initiating the visualization modules.

2. THE BRIDGES SYSTEM

As illustrated in Fig. 1 the BRIDGES system is based on a client-server model; the client consists of the needed infrastructure for students to implement all of the basic data structures (currently Java and C++ APIs are supported). At any point, a representation of the constructed data structure can be generated and transmitted to the BRIDGES server to generate a visualization. The visualization will be displayed on a specific web link provided to the user. The BRIDGES server provides needed interfaces to data sources that make it easy for the user to make data requests. The acquired data (to be used by the user as part of course projects) is also cached on a Mongo database. Finally the server maintains a gallery of projects created by the user that can be shared. The visualization will appear on the specified web page with some capabilities for interactive manipulation. All projects are held in a database and the client will have access to an interactive gallery under their BRIDGES account.

2.1 An Example BRIDGES Program

We begin with an example template of what a BRIDGES program would look like. Fig. 2 shows the main calls made from an example program and consists of the following steps. The **Bridges** class encapsulates the details of the communication between the client and the server and uses additional helper classes in its implementation.

- **Initialization.** This step creates a BRIDGES class and takes several parameters, including an assignment number, a user id (generated by the user when he/she creates a BRIDGES account) and the corresponding user name. These parameters are used in forming a custom web link for the user's data structure.

- **Data Structure Construction.** In this step, the user constructs and manipulates any of the BRIDGES supported data structures using the BRIDGES client classes (see Section 2.2.1).

- **Specification of Data Structure Type.** This step specifies the handle to the data structure that will be transmitted to the BRIDGES server for visualization. This can be the head of a linked list, root of a tree, graph adjacency list, or array name.

- **Visualization.** This step results in the creation of a JSON string representation of previously specified data structure and its transmission to the BRIDGES server using an HTTP post request. If this is successful, a web link is returned to the user for viewing the visualization. The assignment is also stored in the MongoDB that the user can subsequently access through his BRIDGES account. This step can be repeated any number of times: the data structure can be modified, the handle respecified followed by visualization.

2.2 BRIDGES Design

2.2.1 *BRIDGES Client*

The BRIDGES client consists of a minimal set of building blocks that are needed by students in a typical sophomore level course on data structures: array, list, tree structures, graph. The class structure is illustrated in Fig. 3, roughly follows the description and implementation of Shaffer[16, 15] contains the following components:

Element. This is the foundational class in BRIDGES. Arrays, lists, tree and graph nodes are constructed using this element. Elements have a unique id, a label (used in the visualization), and visual properties (size, shape, opacity, color). Elements can also be related to another element via a *link*, as would be needed for trees, linked lists and graphs. Links have attributes: color, thickness, and opacity. Elements are declared with a generic parameter, that can be used to hold application specific data (Tweet, actor, movie, etc)

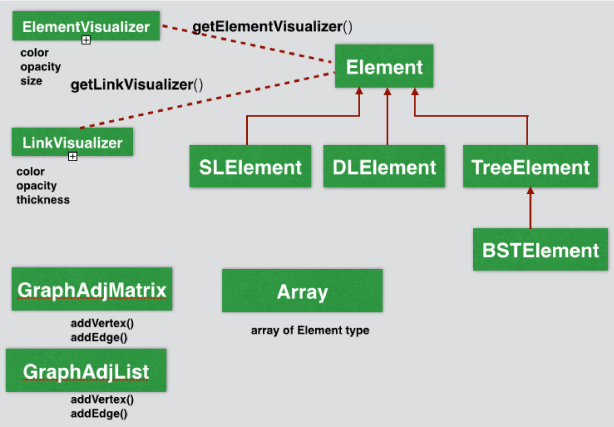


Figure 3: BRIDGES Client. Consists of the basic building blocks to construct data structures, such as arrays, linked lists, binary trees and graphs. The visualizer classes affect the visual attributes (color, opacity, size, thickness) of the elements (nodes) and links, depending on the data structure.

SLElement. Derived from Element, SLelement represents a singly linked element and has a link to the succeeding element, via a *next* pointer.

DLElement. Derived from Element, DLelement represents a doubly linked element and has links to previous and succeeding elements, via *prev* and *next* pointers.

TreeElement. Derived from Element, TreeElement is a binary tree node, with links to its two children, via *left* and *right* pointers.

BSTElement. Derived from TreeElement, augments the TreeNode with *key* value; must be an orderable type.

GraphAdjMatrix. Implemented as a two dimensional hash table, the graph supports vertices indexed by any orderable type (int, float, string, etc).

GraphAdjList. Implemented as a hash table of vertices, with each table entry pointing a singly linked list (of type SLelement);

Array. Arrays represent a list of type Element and support the normal indexing operations of arrays as defined in programming languages; however, each element of a BRIDGES array supports visual attributes.

Implementation. Implementations using the above class structures closely follow the implementation examples in [16, 15]. However, by using the object definitions above, the basic data structure elements can now be augmented with visual attributes that can be exploited by the visualization modules. BRIDGES currently supports both Java and C++ implementations. The Java implementation uses *Java Generics*; while the C++ implementation uses *C++ templates*. Generic implementations make it possible to handle real-world datasets where indexing by string (Twitter or Facebook user name, for example) is needed for graph implementations, bypassing a remapping to integer vertex indices. A second advantage is the ability to incorporate application specific dataset as a generic parameter to the base classes.

2.2.2 *BRIDGES Server*

The BRIDGES server has the following functions:

- **Access to Real-World Data.** As described earlier, APIs to real-world datasets such as social networks (Twitter, Facebook) can be quite complex and beyond the scope of sophomore level CS students. Thus BRIDGES provides an easy to use interface to acquire data from external data sources. To date, implemented interfaces to services include Twitter, RottenTomatoes, and IMDB, and data from Twitter follower lists and timelines from a few public accounts (ieeevis, twitterapi, wired, US geological survey, and earthquake) have been incorporated directly into course projects. Queried data from the various sources is cached in a Mongo database for the sake of efficiency and data reuse.

- **Visualization.** The BRIDGES server is responsible for receiving data structure representations (in JSON string format) from the client and generating a visual representation (students will be provided a web link). All such requests are authenticated, parsed and stored in a database (MongoDB). Uploaded assignments can be accessed with a direct url or via the user gallery and can be made public or private, allowing users to share some visualizations and hide others for later review or modification. Visualizations of arrays, linked lists, binary trees and graphs (node-link diagrams) are supported by BRIDGES.

Implementation. The server-side implementation is a Node.js application, utilizing Jade templating, a MongoDB database, Javascript, and a variety of popular Javascript libraries, such as D3[3], jQuery, and Underscore.js. The database is used to keep a record of all users, projects, and cached datasets.

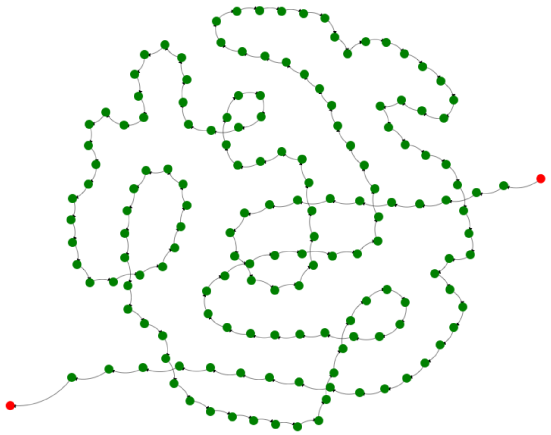
2.3 BRIDGES Intervention in Data Structures Courses

In our CS program, students take the data structures course at the beginning of their sophomore year, followed by an algorithm analysis course. All BRIDGES projects were assigned to students in the data structures course. Typical enrollment in this course is around 50. The BRIDGES intervention was performed in 1 section of the data structures course in fall 2014 and Spring 2015.

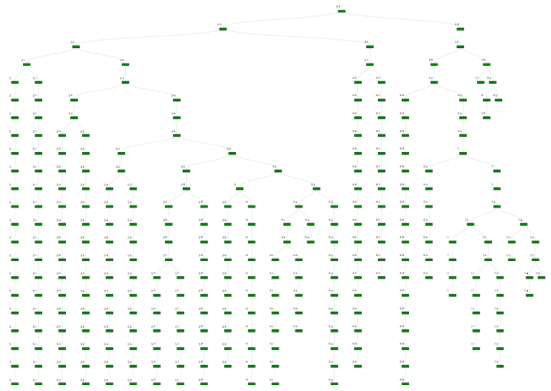
2.3.1 *Fall 2014*

Three BRIDGES projects were assigned in the Fall 2014 semester:

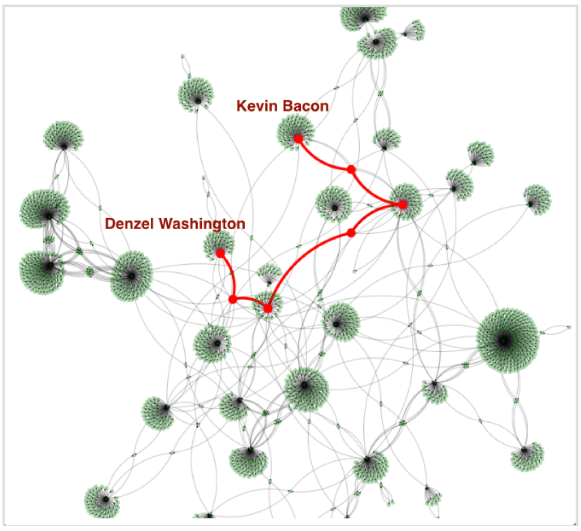
- **Queue:** In this project students to implement and test the Queue ADT, followed by using a live stream of Earthquake Tweet data (US GIS earthquake Tweets). Helper classes were provided to parse the quake data to simplify the data processing tasks. Students had to complete two tasks (a) Given a fixed size queue, incoming tweet data were enqued; if the queue became full, then the oldest tweets were dequed and snapshots of the queue visualizations were to be demonstrated, (b) Given a fixed size queue, incoming data items were to be filtered by quake magnitude prior to entering



(a) Singly Linked List of IMDB dataset containing 1815 actor-movie pairs. Each node is a unique actor; mousing over a node (not shown) displays a list of that actor's movies.



(b) Binary Search Tree of Earthquake Tweet Dataset (US GIS), ordered by quake magnitude.



(c) Bacon Number Computation. Actor-Movie Graph of an IMDB dataset containing 1815 actor-movie pairs. Results of the Bacon number computation between Denzel Washington and Kevin Bacon. The red nodes and links display the path from Denzel Washington to Kevin Bacon.

Figure 4: BRIDGES Project Examples

the queue. Various thresholds should be experimented with queue snapshots visualized.

- **Binary Search Tree.** This project continued to use the earthquake Tweet data, but inserted the records (quake magnitude as the search key) into a binary search tree, followed by visualization of the tree structure. Tasks on the search tree included (a) modifying the insertion algorithm to display insertion path, (b) implement the find algorithm and demonstrate (visually) searching for a quake of a particular magnitude. Fig. 4b illustrates the binary search tree sorted by earthquake magnitudes.

- **Graph (Bacon Number Computation).** This project involved building a graph using a reduced version of the IMDB dataset and implementing the Breadth First Search algorithm on a graph to compute the Bacon Number of an actor in an actor-movie graph[14]. The project used a curated IMDB dataset containing 1815 actor-movie pairs. Tasks involved building the actor-movie graph, determining the Bacon Number of any actor in the graph, and highlighting the path from the queried actor to the Kevin Bacon node. Fig. 4c illustrates the actor-movie graph and the path from the actor Denzel Washington to Kevin Bacon.

2.3.2 *Spring 2015*

Four BRIDGES projects were assigned in the course:

- **Singly Linked List.** The IMDB actor-movie dataset was used in this project. Students had to read in the dataset and build a sorted linked list of the unique set of actors; the data field of each node would contain the list of movies corresponding to the actor. Tasks included finding a specific actor, followed by highlighting the node (if found), adding in new actor-movie pairs and removing an actor. Fig. 4a illustrates the singly linked list sorted by actor names.

- **Stacks: Expression Evaluation.** This project required students to build a linked list-based stack using BRIDGES elements. The stack was then used to evaluate expressions. Bridges visualizations were used to display the contents of the stack after each operation.

- **Binary Search Tree.** This project was similar to the project from the fall 2014 with slight changes in the required tasks.

- **Graph (Bacon Number Computation).** Identical to the Fall 2014 graph project.

3. EVALUATION

We have used BRIDGES in one section of our Data Structures course (ITCS 2214) for the past 2 semesters; the projects that used BRIDGES are described in Sections 2.3.1,2.3.2. The enrollment in each of these 2 sections was capped at 55; the enrollment stabilized at 36 in the Fall 2014 semester and 32 in the Spring 2015 semester. Each BRIDGES project was followed by a project survey with 15 questions relating to the concepts learned in the assignment, the required programming knowledge relative to their own experience, time required for completion of assignment, sources of support

Table 1: Responses to "The assignment was relevant to my career goals" for assignments 1-4.

	#	#Resp	SA%	A%	N%	D%	SD%
Fall 2014	1	29	0.0	48.2	20.7	31.0	0.0
	2	28	17.9	28.6	28.6	17.9	7.1
	3	19	15.8	52.6	15.8	10.5	5.3
Spring 2015	1	27	14.8	14.8	55.6	7.4	7.4
	2	24	29.1	37.5	29.1	4.1	0.0
	3	27	25.9	37.0	29.6	3.7	3.7
	4	28	35.7	25.0	25.0	7.1	7.1

Table 2: Responses to "The assignment was trivial and not essential to learning about computing." for assignments 1-4.

	#	#Resp	SA%	A%	N%	D%	SD%
Fall 2014	1	29	3.5	17.2	24.1	44.8	10.3
	2	28	3.6	10.7	17.9	60.7	7.1
	3	19	5.3	10.5	15.7	42.1	26.3
Spring 2015	1	27	7.4	11.1	25.9	40.7	14.8
	2	24	0	0	25.0	41.6	33.3
	3	27	7.4	0.0	14.8	44.4	33.3
	4	28	10.7	3.6	25.0	21.4	39.2

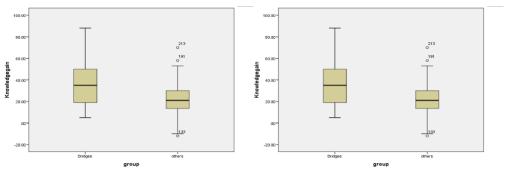
(mentors, TAs, external sources), relevance of the assignment to career goals, difficulty of the assignment, and degree to which the assignment increased their interest in computing. Responses used a 5 point Likert scale, ranging from strongly agree to strongly disagree. Tables 1,2, and 3 detail the responses to the survey questions, (1) *"The assignment was relevant to my career goals"*, (2) *"The assignment was trivial and not essential to learning about computing."*, and (3) *"The assignment increased my interest in computing"*.

- *"The assignment was relevant to my career goals"*. In the fall 2014 surveys (Table 1), a majority of students agreed on this question, on all 3 projects (48% vs. 31%, 46% vs. 25%, 68% vs. 16%) with the remainder being neutral. In the spring semester, the results were even stronger (29% vs. 14%, 66% vs. 4%, 63% vs. 7%, 61% vs. 14%), except for the first project where there were a large number of neutral responses (55%). This was a more difficult project that should have been assigned in multiple parts; the subsequent projects were corrected, so that students found the tasks more manageable.

- *"The assignment was trivial and not essential to learning about computing"*. In the fall 2014 surveys (Table 2), a majority of students disagreed (55% vs. 38%, 68% vs. 14%, 68% vs. 26%) for all projects and in the spring surveys the results were even stronger (55% vs. 18%, 75% vs. 0%, 77% vs. 7%, 61% vs. 14%).

Table 3: Responses to "The assignment increased my interest in computing" for assignments 1-4.

	#	#Resp	SA%	A%	N%	D%	SD%
Fall 2014	1	29	0.0	20.7	44.8	24.1	10.3
	2	28	10.7	25	32.1	21.4	10.7
	3	19	5.3	42.1	31.6	15.8	5.3
Spring 2015	1	27	3.7	22.2	40.7	14.8	18.5
	2	24	20.8	45.8	25.0	8.3	0.0
	3	27	14.8	48.2	25.9	7.4	3.7
	4	28	21.4	32.1	21.4	17.9	7.1



(a) Fall 2014 (b) Spring 2015

Figure 5: Knowledge Gains using BRIDGES

- *"The assignment increased my interest in computing"*. In the fall 2014 surveys (Table 3), the results were mixed, with a minority agreeing on project 1 (21% vs. 34%), slight majority on project 2 (35% vs. 32%), and a majority on project 3 (47% vs. 24%). In the spring 2015 surveys, a minority agreed on project 1 (26% vs. 33%), a majority agreed on the remaining 3 projects (66% vs. 8%, 63% vs. 11%, 53% vs. 25%). As mentioned earlier, project 1 was not modularly designed to help students to complete subtasks as part of the larger assignment, resulting in frustration for many students.

Finally, there was a major version change in BRIDGES between the two semesters. Version 2 of BRIDGES was much more robust, with an improved API (Fig. 3) that was easier to use; additionally, the documentation and tutorials were also vastly improved. Thus, the improvements in student responses across the two semesters were within our expectations.

Knowledge Gains. We also evaluated the knowledge gains of students using BRIDGES compared to students in the remaining sections of data structures. For this we created a knowledge test in data structures with contributions from all instructors teaching the course. A question bank of about 105 multiple choice questions and 38 short answer questions was created. An external project evaluator chose a subset of these questions (35 multiple choice and 6 short answers) to be used for pre/post tests, administered at the beginning and end of the semester. Students from four sections of the fall 2014 data structures course and 4 sections of the Spring 2015 underwent the the knowledge test; 1 section used BRIDGES and the other 3 did not.

Fig. 5 illustrates the knowledge gains using the pre-post scores from the knowledge test. The box plot in Fig. 5 shows knowledge gains from pre-test to post-test, with significant gains for both the BRIDGES section and the control group. It is apparent that the BRIDGES group showed larger gains ($M = 39.12$, $SD = 14.6$) than the control group ($M = 22.\ 00$, $SD = 13.28$), $t\ (93) = 5.33$, $p < .001$. However, each section was taught by a different instructor. As such, the differences in knowledge gains for the groups could be explained at least partially by the differences in instructor emphasis on the knowledge tests; the BRIDGES instructor used part of the knowledge test for the final exam while the other instructors used it as a classroom exercise that did not count toward the final grade. Therefore, it is not possible to definitively pinpoint BRIDGES as a reason for increased knowledge gains.

4. RELATED WORK

Closely related to our work are the approaches in Buckley et al.'s *Socially Relevant Computing* [6] and Bart et al.'s *RealTime Web*[2]; Buckley's work incorporates real world scientific applications into both their introductory and senior capstone courses to make them more interesting and relevant. Bart's *RealTime Web* provides a set of flexible client libraries to request, parse and return real-time data from a number of web sources (Yelp, weather reports, Yahoo Finance, etc.) Our approach goes even further to make the course material relevant; in addiiton to providing easy access to real-world datasets, we provide instantaneous visualizations of the data structures that are built by the students themselves that can help to improve the understanding of the data structure and the algorithms that operate on them.

Visualizations have long been promoted as a way to improve student understanding of data structures and algorithms [1, 12, 5], and the research literature highlighting the ability of interactive visualizations to enhance discovery is vast. Previous efforts to increase engagement have shown promise for the use of visual programming (e.g., Scratch and Alice) [13, 10] for making the first programming steps easier and more engaging. In addition to providing a graphical interface for piecing together programs, these systems let students build graphically interesting programs and encourage them to explore, experiment, and play. Formal evaluations of Alice[11] have shown increased performance and retention in the programing courses and improved attitudes toward computing, especially for at-risk students.

5. CONCLUSIONS

We have presented BRIDGES, a system that provides students with an easy-to-use API for accessing real-world datasets for use in data structures course projects and the ability to explore interactive visualizations of the data structures that are created as part of a BRIDGES-enabled assignment. Rather than look at (uninteresting) textual output of data structures assignments for verification/debugging, students can use BRIDGES to interact with visualizations of the execution of their assignments on real-world data, with the added ability to share their visualizations with peers, friends, and family. Early results of using BRIDGES over 2 semesters in 1 section of our data structures course are promising, with students reporting increased interest in computing after completing assignments.

Currently, we are extending BRIDGES on 3 fronts:

- **External Data Sources.** Ultimately, we expect that BRIDGES will provide access to large variety of interesting external data sources. We also hope to integrate the rich work of *RealTime Web*[2] to complement BRIDGES.
- **Peer Mentoring.** An important component of this project is to build strong connections within the major through peer mentoring, using BRIDGES as a shared interest; specifically, we plan to pair students in data structures courses with the students in the senior-level software development capstone course who are tackling software development projects to extend BRIDGES.
- **Extended Evaluation.** BRIDGES is open source and is available at **http://bridgesuncc.github.io/**. Java and C++ versions are available for deployment by instructors. As more instructors adopt BRIDGES, we plan to perform a larger scale evaluation that measures computing attitudes and engagement in data structures and algorithms courses across institutions.

6. ACKNOWLEDGMENTS

This work was supported by a grant from the National Science Foundation, DUE-1245841.

7. REFERENCES

[1] R. Baecker. Sorting out sorting: A case study of software visualization for teaching computer science. *Software visualization: Programming as a multimedia experience*, 1:369–381, 1998.

[2] A. C. Bart, E. Tilevich, S. Hall, T. Allevato, and C. A. Shaffer. Transforming introductory computer science projects via real-time web data. In *Proc. of the 45th ACM Technical Symposium on Computer Science Education*, pages 289–294, 2014.

[3] M. Bostock, V. Ogievetsky, and J. Heer. D3: Data-driven documents. *IEEE Trans. Visualization & Comp. Graphics*, 2011.

[4] J. D. Bransford, A. L. Brown, and R. R. Cocking. *How people learn: Brain, mind, experience and school*. National Academy Press, 1999.

[5] M. H. Brown and R. Sedgewick. *A system for algorithm animation*, volume 18. ACM, 1984.

[6] M. Buckley, J. Nordlinger, and D. Subramanian. Socially relevant computing. In *Proc. of the 39th SIGCSE Technical Symposium on Computer Science Education*, pages 347–351, 2008.

[7] J. Cohoon. Just get over it or just get on with it. In *Women and Information Technology: Research on Under-Representation*. MIT Press, 2005.

[8] J. Cohoon and L.-Y. Chen. Migrating out of computer science. *Computing Research News*, 15(2), 2003. http://archive.cra.org/CRN/articles/march03/cohoon.chen.html.

[9] Computing Research Association. CRA Taulbee Survey 2010-2011, 2010-2011.

[10] W. P. Dann, S. Cooper, and R. Pausch. *Learning to Program with Alice*. Prentice Hall, 2005.

[11] B. Moskal, D. Lurie, and S. Cooper. Evaluating the effectiveness of a new instructional approach. In *Proc. of the 35th SIGCSE Technical Symposium on Computer Science Education*, pages 75–79, 2004.

[12] W. C. Pierson and S. H. Rodger. Web-based animation of data structures using jawaa. In *ACM SIGCSE Bulletin*, volume 30, pages 267–271. ACM, 1998.

[13] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, and Y. Kafai. Scratch: programming for all. *Communications of the ACM*, 52(11):60–67, 2009.

[14] R. Sedgewick and K. Wayne. *Introduction to Programming in Java*. A Case Study: Small World Phenomenon (http://introcs.cs.princeton.edu/java/home/).

[15] C. Shaffer. *Data Structures and Algorithm Analysis in C++*. Dover Publications, 2011.

[16] C. Shaffer. *Data Structures and Algorithm Analysis in Java*. Dover Publications, 2011.