




BRIDGES: Real world data, assignments and visualizations to engage and motivate CS majors

David Burlinson¹ · Matthew McQuaigue¹ · Alec Goncharow¹ · Kalpathi Subramanian^{1,2}  · Erik Saule¹ · Jamie Payton² · Paula Goolkasian³

Received: 1 July 2022 / Accepted: 30 May 2023

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2023

Abstract

BRIDGES is a software framework for creating engaging assignments for required courses such as data structures and algorithms. It provides students with a simplified API that populates their own data structure implementations with live and real-world data, and provides the ability for students to easily visualize the data structures they create as part of routine classroom exercises. The objective is to use the infrastructure to promote a better understanding of the data structure and its underlying algorithms. This report describes the BRIDGES infrastructure and provides evaluation data collected over the first five years of the project. In the first 2 years, as we were developing the BRIDGES projects, our focus was on gathering data to assess whether the addition of the BRIDGES exercises had an effect on student retention of core concepts in data structures; and throughout the 5-year duration of the project, student interest and faculty feedback were collected online and anonymously. A mixed method design was used to evaluate the project impact. A quasiexperimental design compared student cohorts who were enrolled in comparable course sections that used BRIDGES with those that did not. Qualitative and quantitative measures were developed and used together with course grades and grade point averages. Interest and relevance in BRIDGES programming assignments was assessed with additional survey data from students and instructors. Results showed that students involved in BRIDGES projects demonstrated larger gains in knowledge of data structures compared to students enrolled in comparable course sections, as well as long-term benefits in their performance in four follow-on required courses. Survey responses indicated that some investment of time was needed to use BRIDGES, but the extra efforts were associated with several notable outcomes. Students and instructors had positive perceptions of the value of engaging in BRIDGES projects. BRIDGES can become a tool to get students more engaged in critical foundational courses, demonstrating relevance and context to today's computational challenges.

✉ Kalpathi Subramanian
krs@charlotte.edu

Extended author information available on the last page of the article

Keywords Real-world data · Data structures · Algorithms · Visualization · Engagement · Programming assignments

1 Introduction

Over the last 10 years, enrollments in computer science undergraduate degree programs in the United States has more than tripled (Computing Research Association, 2018, 2019) with corresponding increases in CS Bachelor's degrees. This is indeed welcome, given the the national need for graduates that can contribute to a robust and innovative twenty-first century workforce. At the same time, it is also critical for academic programs to ensure retention of majors by providing a curriculum that is engaging and relevant.

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 (Bransford et al., 1999) and has a positive effect on the retention of computing majors, particularly for women students (Cohoon, 2005). In practice, however, there is little support for educators who want to incorporate this kind of approach into introductory computer science courses. With appropriations for higher education continuing to decline (SHEF, 2019) (with tuition increases to compensate somewhat) and enrollments continuing to rise (Computing Research Association, 2019), departments often look to scale up the class size to meet student needs within the budget constraints. Instructors have more students to guide, assignments to grade, and additional course assistants to coordinate, leaving them with less time to invest in developing new engaging course materials or to pursue more modern pedagogies. The sophomore level courses such as data structures and algorithms (that we focus on in this work) are critical to building a solid foundation for CS students; however, students in these courses do not always see the connection between their own everyday experiences with existing information systems that address real-world problems, and the routine assignments/problems they encounter in course assignments, usually with small or contrived datasets.¹

We introduced in our earlier work (Burlinson et al., 2016) the BRIDGES system and its role in early CS courses (Strahler et al., 2020; Beckman et al., 2020) 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. Our goal with BRIDGES was to make the experience of sophomore level students more relevant and engaging, increase their retention of key foundational knowledge through a careful selection of BRIDGES based course projects, and, thereby increase their chances of continuing in the major. In our earlier work we analyzed BRIDGES interventions on two semesters of a data structures course at

¹ While the authors could not find a proper academic study to back this claim, it frequently appeared in confidential student evaluations. Also that statement is consistent with Situated Learning, and the MUSIC model of engagement.

our institution, and showed improved knowledge gains by comparing the BRIDGES group to the remaining sections of the data structures course sections. Project surveys assessing the use of BRIDGES were largely positive.

To date, BRIDGES has been used for over 5 years in data structures courses at about a dozen institutions in the United States, spanning universities, four year colleges, community colleges, and, more recently, high schools (AP CS courses) and it continues to be actively used. The software has been used by over 1000 students since its inception. Over the years, the toolkit and API have undergone several major revisions to improve its robustness and performance. New features have been introduced together with expanded support for the toolkit to be used with C++ and Python programming languages (the earlier work supported only Java). The team has developed more detailed documentation, tutorials, and assignment descriptions to provide instructors with the additional support to demonstrate and connect core CS concepts to real-world applications.

In this work, we present the current design and architecture of BRIDGES (Section 3.2), and example projects that can be enabled through the use of framework in data structures and algorithms courses. Our objectives are two-fold, (1) assess the retention of core concepts in data structures through of use BRIDGES exercises, and (2) demonstrate long-term gains by tracking the performance of student cohorts in follow-on core courses. We use a mixed-methods design to evaluate the impact of the project. Qualitative and quantitative measures were developed and used in conjunction with course grades and grade point averages for the evaluation. Additional survey data from students and instructors were used to gauge interest and relevance of the newly developed learning materials. Specifically, we detail both immediate (Section 4.1) and long-term student knowledge gains (Section 4.2) and student and instructor feedback (Sections 4.3, 4.4) gathered over the past five years.

2 Related work

In this section we present work that relates to student interest and motivation, existing assignment collections that target student engagement, use of visualizations and real-world data in CS curriculum.

Engagement is closely tied to student motivation, and there is a host of work on understanding student motivation. Jones proposed the MUSIC model for instructors to consider in their course design, which consists of **e**Mpowerment, **U**sefulness, **S**uccess, **I**nterest, and **C**aring (Jones, 2009). Instructors are recommended to incorporate some or all of these into their instruction. Blumenfeld and his colleagues (Blumenfeld et al., 2006, 1991) focus on other determinants of motivation and cognitive engagement, such as Value, Competence, Relatedness, and Autonomy as factors that can help engage students. For instance, students that see value and relevance in course content exhibit increased efficacy, sense of belonging and flexibility in pursuing their learning goals. The usage of real and live data and of visualization in BRIDGES aims at enabling students to see value and relevance in their course work. In the MUSIC parlance, BRIDGES assignments target Usefulness by enabling students to work on real world

application, Interest by integrating data set from domains they may care about, and eMpowerment by enabling students to customize their assignments.

In recent years, active learning techniques have been implemented in classrooms to promote student engagement, and can include any combination of lab-based instruction, flipped classroom settings, gamification, peer-learning, or use of multimedia content (Pirker et al., 2014; Guzdial, 2003; Horton et al., 2014; Latulipe et al., 2015). These methods combine interesting curriculum content as well as more active participation of learners (eg., collaboration, hands-on work, peer teaching, rapid feedback) in improving the student experience in the classroom. Another prominent project is the POGIL (Process Oriented Guided Inquiry Learning), which combines active learning, collaboration and team work through highly structured set of activities to promote engagement and learning. POGIL was first introduced in Chemistry courses and has gained acceptance in CS over the years (Kussmaul, 2012), especially in early courses (VanDeGrift, 2017; Hu & Kussmaul, 2021); however, it does require significant effort to transition from a traditional to a POGIL model. The goal of BRIDGES remains the same, in terms of increasing student engagement and motivation, but accomplishing it through *delivered content, by bringing in real-world relevance of CS to the classroom and visualizations of students' own work*, as part of routine class exercises.

The goal of courses and curriculum is the overall education and academic success of students. To that end, materials that capture the imagination of incoming students and reinforce their interest and motivation in computing are particularly valuable. This is a common thread in popular assignment repositories, such as Nifty Assignments (Parlante, 2018), Peachy Parallel Assignments (NSF/IEEE-TCPP Curriculum Initiative, 2018-2022), ModelAI (AAAI, 2018), Groovy Graphics Assignments (Groovy Graphics Assignments, 2023) and EngageCSEdu (Monge et al., 2015; NCWIT, 2018), all of which contain the 'fun' factor and some relevance to real-world data. Nifty assignments always include aspects of student engagement as part of the assignment review (Parlante, 2021), as do game-themed assignments (Drake & Sung, 2011; Sung et al., 2008). Layman reported a study that analyzes a collection 200 assignments in CS1 and Software Engineering courses from over 70 CS programs (Layman et al., 2007). They classified the CS1 assignments that 1) were games, 2) had little practical context (not games), 3) had practical context, and 4) had social relevance. Their study found that over 40% of the assignments had no practical or meaningful context, with the rest distributed between games and those with a practical context. A key factor that distinguishes our approach is the integration of many of these engagement factors (use of real-world data integrated with the assignment, data structure visualizations, interactive games), while not being distracted from using external tools to accommodate them; for instance, assignments that incorporate visualizations or images typically require using an external toolkit, or are restricted to a specific language, limiting their usage.

Related to our work are the engagement approaches that are classified under the category of *Socially Relevant Computing*; Buckley's work incorporates real-world scientific applications into both their introductory and senior capstone courses to make them more interesting and relevant (Buckley et al., 2008). 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, such as Yelp, weather reports, Yahoo Finance, etc. (Bart et

al., 2014). The use of media (audio, images, movies) computation in early CS courses (Guzdial and Ericson, 2016) and the use of interdisciplinary projects (Havill, 2021) are examples of using more engaging projects in the classroom. Our approach goes even further to make the course material relevant; in addition to providing easy access to real-world datasets, BRIDGES facilitates instantaneous visualizations of the data structures that are built by the students themselves, that can help in the understanding of the data structure and the algorithms that operate on them.

Visualizations have long been promoted as a way to deepen student understanding of data structures and algorithms (Baecker, 1998; Pierson and Rodger, 1998; Brown and Sedgewick, 1984); Previous efforts to increase engagement have shown promise for the use of visual programming (e.g., Scratch and Alice) (Resnick et al., 2009; Dann et al., 2005) 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 (Moskal et al., 2004) have shown increased performance and retention in the programming courses and improved attitudes toward computing, especially for at-risk students.

3 The BRIDGES system

At its core, BRIDGES is a toolkit that enables students in early CS courses (CS1, CS2, Data Structures, Algorithm Analysis) to implement routine course projects assigned by the instructor by providing easy to use functional interfaces to external datasets that can be used as part of an assignment. Secondly, rather than just use console output, BRIDGES provides the capability to build and display visualizations of the data structures that students implement as part of their assignment. Figure 1 illustrates the core of a BRIDGES student program. It consists of creating the BRIDGES object, extracting an external data set through a function call, building the data structure, and incorporating attributes of the data into the data structure. The data structure can then be visualized using a function call. Figure 1 also illustrates some example data sets and visualizations supported by BRIDGES, such as IMDB actor/movie data in a graph application using Breadth First Search (BFS), OpenStreet map data used in a graph application, USGS earthquake Tweet data used in a binary search tree, and an image represented using a K-D (spatial search) tree.

3.1 An example BRIDGES program

Figure 2 provides an example of a BRIDGES program, showing the main calls and 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 object and takes several parameters, including an assignment number, a user id (generated by the user when he/she creates a BRIDGES account) and an API key for authentication. These parameters are used in forming a custom web link for the visualization.

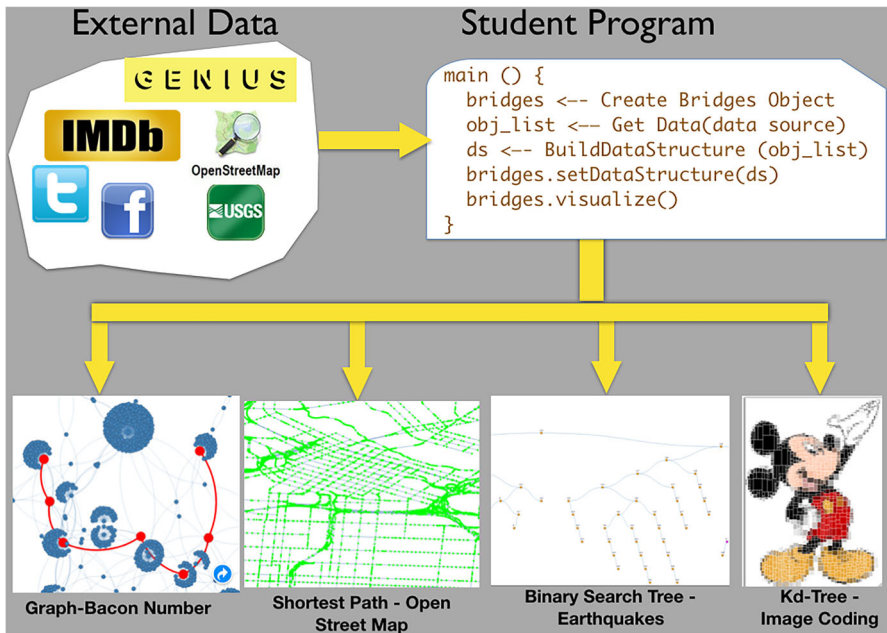


Fig. 1 BRIDGES System Overview. A user constructs programs typical of early CS courses (CS1/CS2/Algorithms/Data Structures) using the BRIDGES client classes (Fig. 3) in Java, C++ or Python. The representation of the output (data structure, game grid, etc.) is sent to the server by the client and stored in a database after parsing and authentication. The output visualization is then displayed on a web browser with a specific link provided to the user. Depending on the assignment and user requests, external data may be extracted from a number of different sources by the server, as illustrated above

- **Data Structure Construction.** In this step, the user constructs and manipulates any of the BRIDGES supported data structures using the BRIDGES client classes as part of their programming assignment (see Section 3.2.1).
- **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, array object, or a symbol collection.
- **Visualization.** This step results in the creation of a representation of the previously specified data structure (implemented using a JSON string) and its transmission to the BRIDGES server using an HTTP post request. If this is successful, the assignment is then stored in a database (a Mongo DB in the current implementation) and a custom web link is returned to the user for viewing the result. The assignment can also be viewed from the user's project gallery at any time. This step can be repeated any number of times: the data structure can be modified, the handle respecified, followed by visualization. Also, a sequence of visualizations can be generated from a BRIDGES program and these will be displayed as an array of

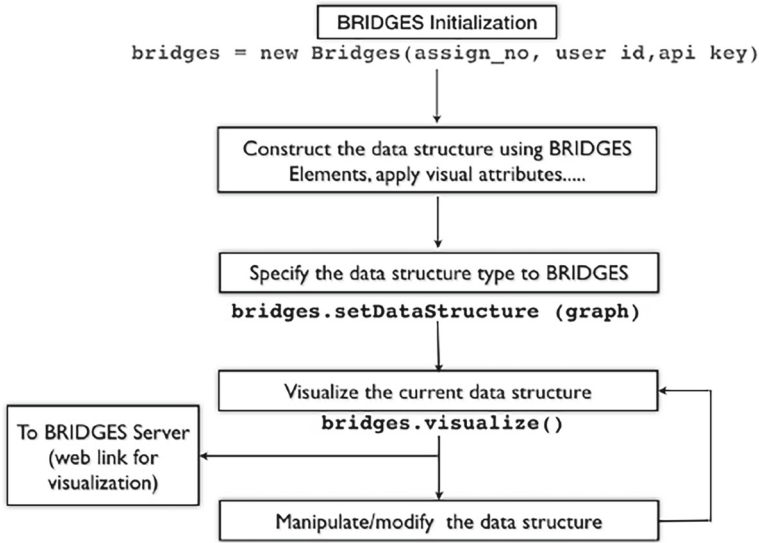


Fig. 2 An Example BRIDGES program. Consists of an initialization step to specify assignment number, user name and api key (for authentication). This is followed by user's construction of the data structure using BRIDGES client classes and specifying visual attributes. The final 2 steps involve specifying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

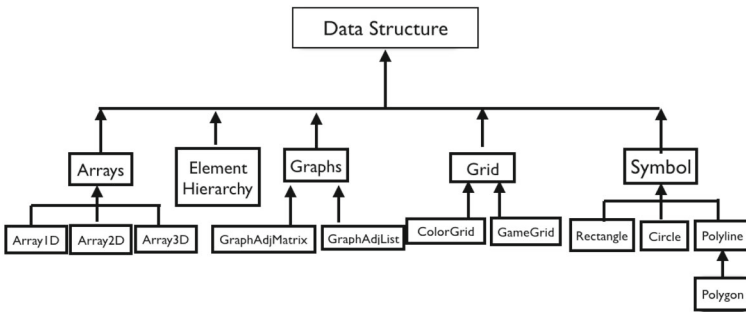
visualizations. For example, insertion of multiple values into a binary search tree can be viewed by generating a visualization after each value is inserted into the tree.

3.2 BRIDGES design

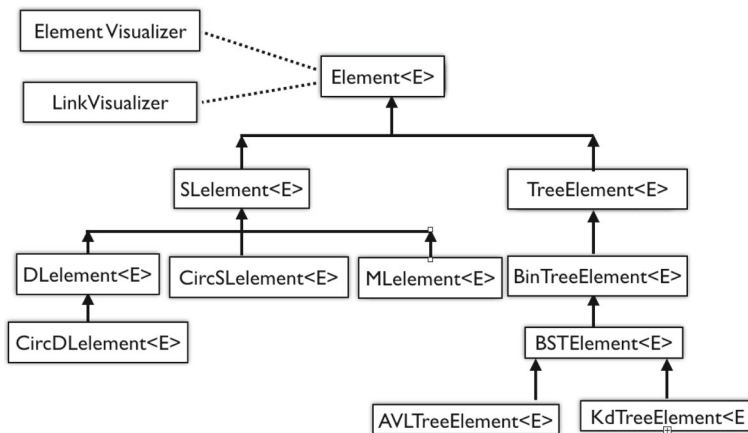
3.2.1 BRIDGES client

The BRIDGES client consists of a set of building blocks that are needed by students in early CS courses to complete typical programming assignments. We currently support assignments for CS1, CS2, Algorithm Analysis, and Data Structures courses, with bindings for Java, C++, and Python. As seen in Fig. 3a, BRIDGES provides support for arrays, lists, tree structures, graphs, game grid (for building simple interactive games), a symbol collection (for drawing simple shapes) and plotting tools. Central to BRIDGES is the element hierarchy, illustrated in Fig. 3b. The Element hierarchy provides the basic node structures for many of the data structures, e.g., linked list node, binary tree node, graph structure with methods to add vertices and edges, etc. The Element hierarchy roughly follows the structure of the nodes described in Cliff Shaffer's textbooks (Shaffer, 2011a, 2011b).

- Element** This is the foundational class in BRIDGES. Arrays, lists, tree, and graph nodes are constructed using instances of Element. Elements have a unique id, a label, and visual properties, such as size, shape, opacity, and color. An element can



(a) BRIDGES Object Hierarchy. BRIDGES supports Arrays (1D, 2D and 3D), list and tree structures, graphs (adjacency list and adjacency matrix), a Grid type useful for signal/image based assignments and symbol collections that supports construction of shapes for representing problems (for instance, Towers of Hanoi problem).



(b) BRIDGES Element Hierarchy. Consists of the basic building blocks to construct different types of lists and tree structures. Lists include single and doubly linked lists, circularly linked lists, and multilist. Tree structures include general tree (with arbitrary number of children), binary tree, binary search tree, AVL tree and KD Tree. Each element class provides the basic data and methods to construct a single element of the data structure. Elements also contain two additional classes for visualization: ElementVisualizer and LinkVisualizer. These provide visual attributes (color, opacity, size, thickness) that can be used to enhance the final visualization, for example, draw the shortest path in a graph, highlight key nodes that satisfy certain criteria.

Fig. 3 BRIDGES Object Hierarchy

also be linked to another element, as would be needed for trees, linked lists and graphs. Links have attributes such as color, thickness, opacity, and label. Elements are declared with a generic parameter, that can be used to hold application specific data (Tweet, actor, movie, earthquake, book, game, etc.). All subclasses inherit the visual attributes held in Element.

- **List Elements.** A number of different types of lists are supported, as seen in the element object hierarchy in Fig. 3b. SLElement, DLElement support single and doubly linked list elements, while CircSLElement, CircDLElement are for supporting circular linked lists. MLElement supports multilists.
- **Tree Elements** A number of tree structures are supported in BRIDGES. At the highest level we have TreeElement, which is a general tree node with an arbitrary number of children. Derived from the tree element are the BinTreeElement and BSTElement, for binary trees and binary search trees, respectively, the latter adding a key value to the element. The AVL tree element adds attributes to support a balance factor and tree height, while the KDTreeElement supports spatial search structures with a partitioning dimension attribute.
- **Graphs.** Graph implementation in BRIDGES includes both adjacency list and adjacency matrix (GraphAdjList, GraphAdjMatrix) representations. The adjacency list uses the singly linked elements (SLElement) to represent the lists and maps (hash tables) to access the adjacency lists of a graph node in constant time. The adjacency matrix implementation is similar, except that it uses maps to access any graph node in constant time (two dimensional map). Graphs use generic parameters for the key, permitting any orderable type (int, float, string, etc.) to be used as a key value to access graph nodes.
- **Array.** Arrays (Array1D, Array2D, Array3D) 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. Currently, 1D, 2D and 3D arrays are supported
- **Symbol.** This class supports a set of simple shapes for drawing arbitrary shapes as part of an assignment (for instance, Towers of Hanoi), or for any assignment that lets students build arbitrary shapes as a warm up and engaging exercise early in the course. Currently, BRIDGES supports Circle, Rectangle, Polyline, Polygon and a Text Label (to annotate shapes). Shapes have location information and can be transformed (translate, scale, and rotate) and grouped to build complex shapes.
- **Grid.** This class is the basis for assignments that operate on a grid structure, such as 1D signals and 2D images. The ColorGrid subclass is for use with 2D images and can form the basis for image related exercises.
- **Visualizers.** Each element supports visual attributes, implemented through two visualizer classes, ElementVisualizer and LinkVisualizer. ElementVisualizer supports attributes for each element and include color, size, opacity and shape. LinkVisualizer provides attributes for links between elements and include color, opacity, thickness and a label.

Implementation. Implementation of the element hierarchy follows the implementation examples in Shaffer (2011a, b). However, by using the object definitions above, the basic data structure elements can now be augmented with visual attributes and used by the visualization modules. BRIDGES API currently supports Java, C++ and Python bindings. The Java implementation uses Java Generics, while the C++ implementation

uses C++ templates. The Python typing structure naturally allows extensions to different types (duck typing). 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 datasets as a generic parameter to the base classes.

The BRIDGES client is also responsible for building a representation of the data structure (we use the JSON format) to be transmitted to the server. This action is performed with each call to visualize the specified data structure. Additional functionality of the client includes preparing and formatting the URL for accessing external data as well as the URL to view the data structure visualization. The client also provides modules for formatting the received data for each external data source in the form of objects (Java, C++, or Python) for ease of use by the user.

3.2.2 BRIDGES server

The BRIDGES server has the following functions:

- **Access to Real-World Data.** Accessing real-world datasets such as social networks (Twitter, Facebook) can be quite complex, require authentication, handling download limits, and possibly require permissions. BRIDGES provides an easy to use interface (typically, a single function call) to acquire the data from external data sources. We have implemented interfaces to a number of datasets, including Twitter, IMDB, OpenStreetMap, NOAA Elevation maps, US geological Survey (earthquake data), Genius API (song lyrics) and Project Gutenberg; these have been incorporated into course projects. Queried data from the various sources is cached in a Mongo database, or a local cache for queries that might be used repeatedly.
- **Visualization.** The BRIDGES server is responsible for receiving data structure representations (in JSON string format) from the client and generating a visual representation; students are provided a web link, when their program executes successfully. All such requests are authenticated, parsed and stored in the MongoDB database. 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, lists, tree structures, graphs, and shape collections 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 (Bostock et al., 2011), jQuery, and Underscore.js. The database is used to keep a record of all users, projects, and cached datasets.

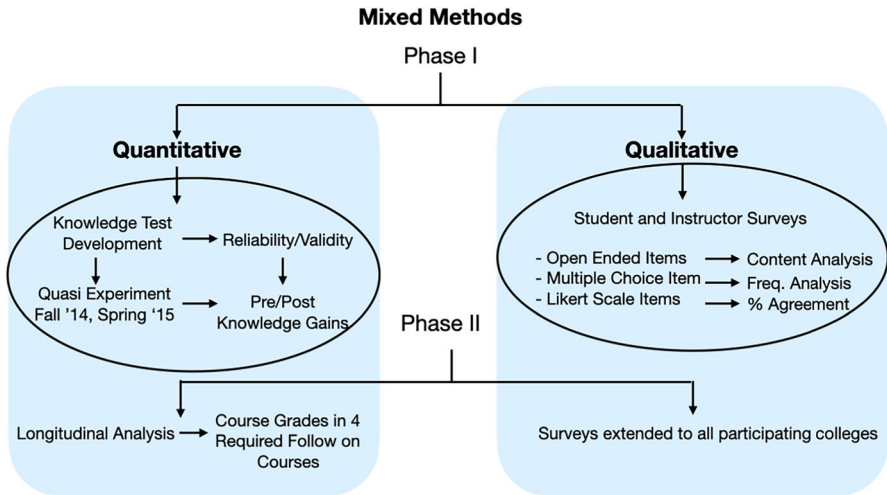


Fig. 4 BRIDGES Project Evaluation Plan

4 Evaluation plan

The funded project collected data over five years and involved two phases using a mixed method design outlined in Fig. 4. The first phase was focused on developing and testing BRIDGES projects on two cohorts of students enrolled in Data Structures at UNC Charlotte during the Fall 2014 and Spring 2015 semesters, while the the second phase extended that testing to other colleges and universities whose instructors agreed to use the BRIDGES infrastructure for class projects.

As we were developing the BRIDGES projects, our focus was on gathering data to assess whether the addition of the BRIDGES exercises had an effect on student retention of core concepts in data structures. Qualitative and quantitative measures were developed and used together with course grades and grade point averages in a quasiexperimental design implemented in one of the sophomore-level data structure courses taught at our university. A knowledge test was developed as an outcome measure and validated to determine how much of the core concepts in data structures were learned and retained. By administering different versions of the test during the first and last weeks of the semester, we could assess immediate gains in knowledge of data structures in the BRIDGES section, in comparison to other sections of the same course that did not participate in the BRIDGES projects. Since there was no coordination among the different instructors of the various sections regarding grading standards, course syllabii, etc., an achievement test specific to data structures' core concepts was essential for the comparison. The comparison group for the BRIDGES section was comprised of all the other sections of the data structures course taught at UNC Charlotte during the Fall 2014 and Spring 2015 semesters. Participation in the BRIDGES project were limited such that the students took the pre/post knowledge

tests; and their instructors assisted with the development and validation of the knowledge test and provided their final course grades. In all other respects, the instructors of the comparison group engaged in their typical teaching techniques, which may or may not have involved homework projects.

Additionally, surveys were developed to collect data online and anonymously from students and instructors who participated in the BRIDGES project. The student surveys administered after each of the projects were completed used open-ended questions to determine if the students could identify the data structure concepts involved in the projects and could reflect on its importance to programming, multiple choice items to determine how long it took to complete the project, and the level of programming required, and Likert scaled items to assess student perceptions of their interest in the project and its relevance to their careers. Instructor feedback was obtained through instructor surveys at the end of each semester; but, also with formative evaluations when instructors were interviewed in advance of each semester to plan course content and project assignments.

The second phase of the project evaluation assessed long-term gains in student performance by comparing the course grades of the students in sections with and without the BRIDGES intervention in four follow-on core CS courses. This plan provided important data about the effectiveness of the BRIDGES intervention on retention of data structure concepts when compared to other sections of the course that did not use the intervention; but it is limited by the fact that there were a number of other sections of the same course, each with their own instructor, and statistical comparisons were of limited value in explaining differences because of unequal group sizes and varying instructor standards and expectations.

Because we extended the use of the BRIDGES infrastructure to other universities and colleges throughout the United States in the second phase of the project, we used the surveys to provide important data about student and instructor perceptions of interest and relevance of the project, relative ease of use, and other relevant issues associated with the experience of using the BRIDGES infrastructure. Students completed a brief BRIDGES assignment survey each time they completed a BRIDGES project, and instructors were asked to complete the Instructor surveys anonymously and online at the end of the semester.

4.1 BRIDGES intervention in data structures courses: Knowledge gains

In our CS program, students take the data structures course at the beginning of their sophomore year, followed by an algorithm analysis course. Typical enrollment in each of the sections of the course is around 50. The BRIDGES intervention was performed in one section of the data structures course in Fall 2014 and Spring 2015 semester. The other sections of the course were used to provide the data for the comparison group. The GPA's of the students who were enrolled in all of the sections were compared prior to their participation in the course. Data are presented in Table 1.

Table 1 Quasiexperimental Results

Measure	BRIDGES Group	Comparison Group	<i>t</i> test	<i>p</i>
<i>Fall 2014 Cohort</i>				
N	54	200		
GPA _{pre-course}	2.21(1.01)	2.76 (1.03)	<i>t</i> < 1	n.s.
Knowledge _{pre}	33.47 (11.6)	32.03 (80.8)	<i>t</i> < 1	n.s.
Knowledge _{post}	72.16 (18.0)	54.87 (14.09)	5.81	0.001
Knowledge _{gains}	36.75 (20.06)	22.57 (15.13)	4.21	0.001
<i>Spring 2016 Cohort</i>				
N	49	113		
GPA _{pre-course}	2.56(0.96)	2.97 (0.72)	2.7	0.009
Knowledge _{pre}	35.55 (10.67)	31.60 (7.79)	2.06	0.044
Knowledge _{post}	74.44 (13.81)	53.76 (13.82)	6.95	0.001
Knowledge _{gains}	39.12 (14.6)	22.00 (13.28)	5.33	0.001

Data in table represent means (SD) and test results. Note. n.s = nonsignificant

4.1.1 BRIDGES Projects

Three BRIDGES projects were assigned in the Fall 2014 semester:

- Queue:** In this project students implement and test the Queue Abstract Data Type, followed by using USGIS Earthquake Tweet data (US GIS earthquake Tweets) as part of a queue application. 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 enqueued; if the queue became full, then the oldest tweets were dequeued 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 the queue. Various thresholds should be experimented with and queue snapshots were to be 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. Figure 5b 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 (Sedgewick & Wayne, 2017). The project used a curated IMDB dataset containing 1815 actor-movie pairs. Tasks involved building the actor-movie graph, determining the Bacon Number (Sedgwick & Wayne, 2020) of any actor in the graph, and highlighting the path from the queried actor to the Kevin Bacon

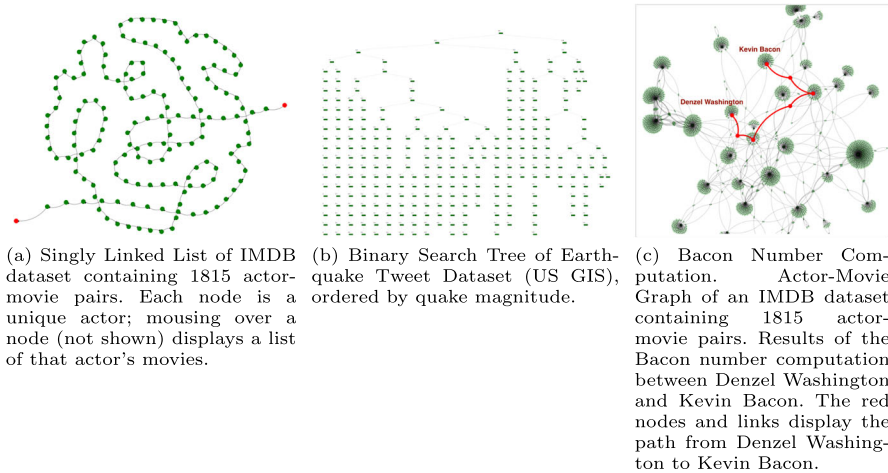


Fig. 5 BRIDGES Project Examples Used in the Course Interventions

node. Figure 5c illustrates the actor-movie graph and the path from the actor Denzel Washington to Kevin Bacon.

Four BRIDGES projects were assigned in the Spring 2015 semester.

- **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. Figure 5a 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.

Figure 5 shows examples of BRIDGES projects used in our interventions.

4.1.2 Development of the knowledge test

The knowledge tests were developed from items submitted that covered one of the 6 core areas of data structures by five data structures course instructors; the 6 areas included Fundamentals, Lists/Stacks/Queues, Binary Trees and Binary Search Trees, Graphs, Heaps, and General: Comparing Data Structures. To establish content validity, the five data structure instructors were asked to rate the appropriateness of each of the items on a 5-point scale (where 1 = not appropriate to 5 = very appropriate). Items were included in the test bank only if the average instructor ratings were four or above. The

final item bank consisted of 105 multiple choice and 43 short-answer items. Test scores were computed by summing the item scores where each multiple-choice item received two points and each short answer 5 points. The item bank was used each semester by a program evaluator to create knowledge tests consisting of 45 multiple-choice and 5 short answer items. Each test included 50 items and was administered by a proctor. Different tests (with items matched in content) were used for the pre/post tests. Item analyses were used to measure test reliability and item difficulty. The results from both semesters showed that the tests had acceptable reliability ($KR20 = .70$) with items that varied in difficulty. Students from five sections of the Fall 2014 data structures course and three sections of the Spring 2015 participated in the knowledge test. Instructors were unaware of the items used in the test prior to its administration. In addition, regression analyses showed that the knowledge test when taken at the end of the semester was found to be strongly related to the final grades of the students in the data structures course, the post knowledge test score accounted for 35% of the variance in Fall 2018 final course grades, and 20% of the variance in Spring 2019 grades.

In order to assess the immediate gains from using the BRIDGES software in the data structures course, we compared the performance gains on the knowledge test for students who participated in the section that included the BRIDGES intervention with those enrolled in sections that did not include the intervention. Table 1 presents the means (SD) of the knowledge tests together with other statistical results. For the Fall 2014 cohort, the students enrolled in the BRIDGES section were not found to differ significantly from the other 4 sections on their GPA at the beginning of the semester nor did the comparison groups differ on their pretest knowledge test scores. However, post-test knowledge scores were found to differ significantly. Data from the Spring 2015 showed some group difference in both pre and post test measures. Pretest GPAs were lower in the BRIDGES group, but the pre knowledge score was higher. Post knowledge score was higher in the BRIDGES Group.

When the post knowledge test scores are subtracted from the pre knowledge test scores for each of the students, we get a measure of the knowledge gains and those data are represented by the box plots in Fig. 6 for both cohorts. Across the semesters, there are significant gains for both comparison groups, and it is apparent that the BRIDGES group showed larger gains than the control group, and for both of these semesters

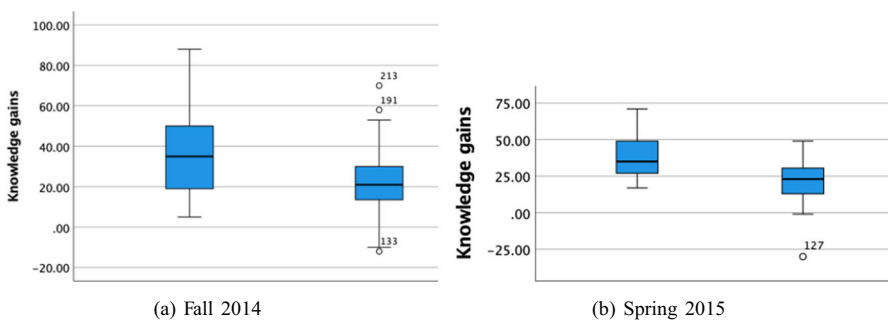


Fig. 6 Knowledge Gains using BRIDGES. Plot on the left is the intervention group and the plot on the right is the control group

the gains in performance on the knowledge test for the BRIDGES group were greater than in the other group. Since each section of the course was taught by a different instructor, the 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. However, the data show that there were immediate gains in retention of the core concepts that were enhanced somewhat by the students who were enrolled in the section with the BRIDGES intervention.

4.2 Longterm gains using BRIDGES: Student progression in the major

Long term gains in student achievement were assessed by comparing the performance of the students enrolled in the sections of the data structures courses with and without the BRIDGES intervention. The longitudinal analysis followed the two cohorts as they progressed through their major in four follow-on required core courses: Analysis of Algorithms, Operating Systems, Software Engineering and Computer Architecture. This approach of looking at the performance of students in follow-on courses has been shown to be an effective indicator for student performance in earlier work (Stroebe, 2016).

Figure 7 compares the percent of students from each of the groups with and without BRIDGES who achieved a grade of C or better in the required follow-on courses in computer science. The grade of C or better was chosen because it was required for progression in the major. It can be seen that for both the Fall 2014 and Spring 2015 cohorts, the percent of students who passed the follow-on course with a grade of C or better was equal to or greater for the students who participated in the BRIDGES intervention. Since data structures is a required prerequisite in all of the follow-on courses,

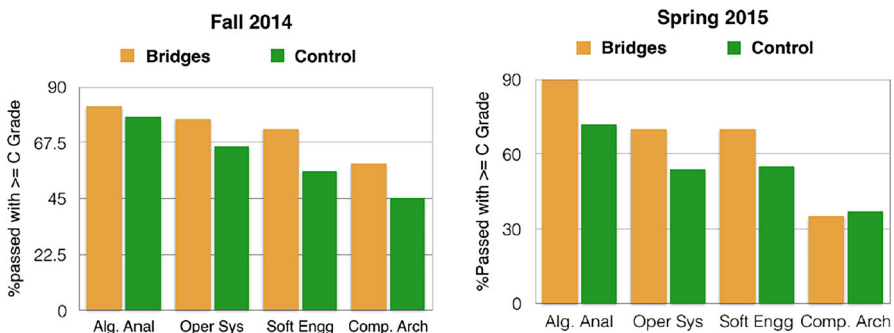


Fig. 7 Comparing long-term student achievement between students who used the BRIDGES toolkit in the Data Structures course vs. Comparison group. The evaluation was performed with 2 cohorts of students (Fall 14, Spring 15). The students were tracked through four follow-on core courses in Computer Science (Analysis of Algorithms, Operating Systems, Software Engineering, and Computer Architecture). BRIDGES cohorts were either the same or outperformed the comparison groups in both Fall 14 and Spring 15 semesters

these data suggest that providing an engaging experience when learning fundamental concepts can have benefits that extend throughout the student's progress through the computer science major.

4.3 Student feedback from project surveys

After completion of each of the BRIDGES homework assignments, students signed on to a Qualtrics web site to fill out the BRIDGES Assignment Survey anonymously. From the students who were enrolled in the BRIDGES intervention from 2014 to 2018, we received 456 responses to the student surveys. Class response rate varied from 50 to 100% participation depending upon the instructor's emphasis placed on the importance of the survey as well as time in the semester that the BRIDGES assignment was completed. Questions consisted of open-ended items that required the students to reflect on concepts learned by the assignment, its significance, and its impact on their programming skills; multiple-choice items that recorded the amount of time spent on the assignment, sources used for help, and items that required use of a 5-point Likert scale to rate relevance of the project, if the project increased their interest in computer science, and the importance of the homework assignment. Appendix A lists all of the items in the student survey.

4.3.1 Qualitative analysis of open-ended items

Student answers to the three open-ended questions (Questions 1, 2, and 3 of Appendix A) were subjected to a content analysis which identified the main themes noted in each of the responses and how often the themes appeared within certain response groups. Two raters, who were senior computer science majors, read the 456 responses made by the students enrolled in the BRIDGES intervention at our university as well as one of 10 other participating colleges and universities. The raters worked independently to classify the student comments according to positive/negative thematic content and to note frequency of occurrence. The ratings from the two raters were averaged and there was 95% agreement among the two raters.

The themes are presented in Table 2 ranked from the most to least frequent for each of the open-ended questions and by positive/negative content. With each, we were interested in comparing responses from our institution's students to those students from other universities and colleges, looking at whether there were any changes as they progressed through the semester with the 1st, 2nd or 3rd BRIDGES assignments. Although there was some variation in the assignments across semesters and between institutions, it is clear from the data that the majority of students were able to clearly identify the essential concept that was learned in each of the assignments. Sixty-seven per cent of the comments for the 1st, 80% of the 2nd, and 85% of the 3rd accurately identified the concept that was under study in each of the assignments. In addition, there was an overwhelmingly positive response (greater than 90%) by our institution's students as well as students from the other institutions when asked to identify why the concept was important and how it improved their understanding of programming. The students suggested that the concepts demonstrated value and usefulness, that they were

Table 2 Content Analysis: Positive and Negative themes from student project surveys over three assignments in data structures courses, from Fall 2014 through Spring 2018 at our institution and a number of external institutions that used BRIDGES. The numbers in the table represent how often each theme occurred in the student responses

	Assignment 1		Assignment 2		Assignment 3			
	Our Inst.	Ext. Inst.	Our Inst.	Ext. Inst.	Our Inst.	Ext. Inst.		
<i>(a) What is the essential concept(s) that was learned by completing this assignment?</i>								
Positive								
Use Linked Lists	33	92	Use Graphs	26	0	Use Search Alg (DFS/BFS)	35	14
Use Queues(1)	20	0	Use Stacks	22	0	Use Graphs	34	0
Use BRIDGES(1)	20	10	Use Trees	16	36	Use Trees	31	16
Visualize Data(1)	7	12	Use Search Alg(DFS/BFS)	16	24	Use BRIDGES	8	16
Data Manipulation	2	0	Use BRIDGES	11	5	Visualize Data	5	0
Use Trees	1	5	Visualize Data	10	11	Use Recursion	5	4
Use Stacks	1	0	Use Recursion	1	11	Use Queues	1	2
Search Twitter	1	0	Use Linked Lists	0	2			
Use External Library	0	1	RunTime Complexity	0	1			
Use Recursion	0	5						
Runtime Complexity	0	6	Unsure	2	0	Unrelated Response	0	1
Not Impressed by BRIDGES	3	2						
Unrelated Response	1	1						

Table 2 continued

Themes	Assignment 1		Assignment 2		Assignment 3	
	Our Inst.	Ext. Inst.	Our Inst.	Ext. Inst.	Our Inst.	Ext. Inst.
<i>(b) Why is this (essential concept(s)) important?</i>						
Positive						
Demonstrates value/usefulness	15	28	27	18	32	4
It is a fundamental Concept	23	32	21	11	20	2
Helps to improve understanding	21	33	15	14	14	6
Provides experience to allow for future use	17	13	12	5	8	5
Provides access to a new tool	2	13	6	3	7	3
Not Sure	6	3	2	0	2	1
Incoherent Response	2	2	2	1	1	0
Required for course	1	0	1	0	1	0
It is not	0	1	0	0	0	0
<i>(c) How does the concept contribute to understanding how to program?</i>						
Positive						
Improves understanding of concept/problem	16	63	24	27	32	9
Provides more tools to choose from	12	12	17	5	18	3
Provides experience for future use	9	4	11	5	9	1
Provides opportunity to apply concept(s)	7	10	7	7	13	1
It is fundamental	10	16	7	4	3	3
Improves efficiency	11	10	6	5	2	1
Provides experience with external code	1	2	6	2	1	0
It does not	2	3	4	0	4	0
Confused	6	3	2	0	2	1
Unsure	1	0	5	1	2	0

fundamental concepts and helped to improve understanding. When asked to identify how the concept contributed to programming, improved understanding of the problem, providing more tools and experience for future use were the prevalent responses. There were only minor variations in responses among different universities and colleges and among the three different assignments.

4.3.2 Rating scale and multiple-choice item responses

The percent of students who agreed that the BRIDGES project increased their interest in computer science, and was relevant or trivial to their career goal (Questions 11, 12, 13 in Appendix A) are presented in Fig 7. These questions were answered using a 5-point Likert scale, which ranged from strongly agree to strongly disagree. Throughout the 5 years that BRIDGES has been under development, a majority of the students agreed that the assignments were relevant and increased their interest in computing, with only a minority expressing the opposite view that the exercises were trivial and not essential to computing. As the software became more stable after the first few semesters of use, and changes were made to address some of the student and instructor issues, there was a noticeable rise in the percent of the students who expressed positive responses to the assignments. The number of students and instructors participating in BRIDGES had noticeably increased over the years and the resulting data are also more stable, as can be seen by the trend in the last 2-3 semesters, with the exception of a unexplainable rise in the percent of students who rated the projects as trivial in the last semester.

Also, noteworthy in the student responses to the project assignment surveys was the amount of time that they reported to complete the BRIDGES assignments. Averaging across all five years of data, we found that 13% indicated spending 3 hrs or less, 31% 4-6 hrs, 24% 7-9 hrs, and 32% reported spending 10 or more hours completing their assignments. Although there was some noticeable variability across the assignments, it is significant that so many of the students reported putting a lot of time into completing each and every one of the BRIDGES projects that were assigned. Moreover, 42% of the students on average indicated that the assignments required more programming experience than they had. Taken together these student responses suggest that the BRIDGES homework assignments required some effort to complete and were perceived as difficult by a number of students (Fig. 8).

4.4 Instructor feedback

Instructor surveys were used to obtain anonymous feedback about the instructor's experiences incorporating the BRIDGES software into their courses. These surveys were used by four instructors who were not part of the BRIDGES development team. The 5-item online survey asked instructors to rate the difficulty of incorporating the software into the course for themselves and their students and to rate the degree to which they thought that it improved the classroom experience. On average, the instructors indicated a moderate amount of difficulty for themselves; but a lot of difficulty for the students. However, the majority (75%) of the instructors noted that it was somewhat easy to incorporate the BRIDGES assignments into the classroom

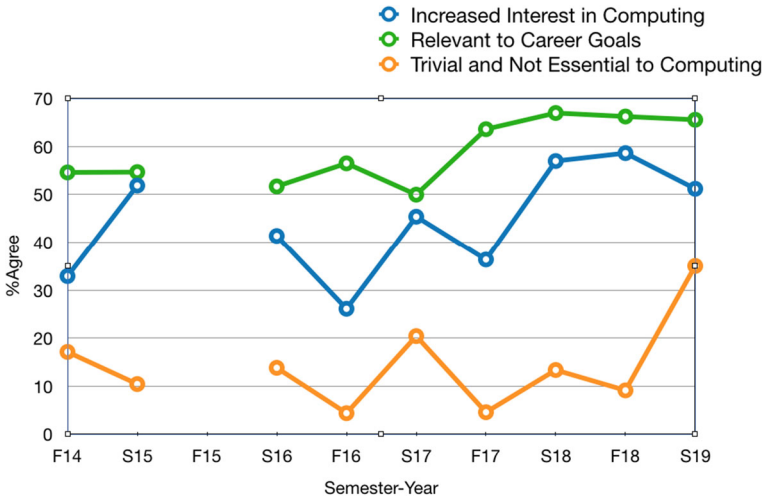


Fig. 8 Student Feedback - Fall 2014 to Spring 2019

lectures and there was consensus from all that it improved the classroom experience for the students. Open-ended comments cited the BRIDGES infrastructure as a strong system with the main benefit as being able to visualize data structures; and when asked to suggest improvements, they requested better documentation at the web site for the students.

5 Discussion

Student and instructor survey responses indicated that some investment of time was needed to use the BRIDGES infrastructure in the data structures course. Instructors required a moderate amount of time to incorporate the projects into their course, while the students needed a greater investment of time to complete the required minimum of three homework assignments using the BRIDGES software. However, the increased efforts spent on completing the BRIDGES assignments were associated with several notable outcomes. Both groups had overwhelmingly positive perceptions of the benefits of engaging in the BRIDGES projects. Students reported positively on the ability to visualize the assignment outputs, working with a API and data sets that peaked their interests in the course topics. Instructors noted that it was easy to incorporate the BRIDGES software into their course and that the classroom experience was improved as a result. Students also understood that the projects were related to their career goals and that it helped to develop their programming skills.

More objective measures showed that the students involved in the BRIDGES intervention showed larger gains in knowledge of the content material in data structures when compared to students enrolled in comparable sections of the course. Perhaps more indicative of the benefits of the BRIDGES intervention for the students, however, is the fact that when you compare the BRIDGES students to those enrolled in

comparable sections of the same course across two semesters, we found that for both cohorts a larger percent of the BRIDGES students passed with a grade of C or better in four of the follow-on courses that were required for progression in the computer science major, suggesting that those students developed to a greater degree the necessary level of CS core concepts and associated programming skills in their foundation course to succeed in the major relative to students enrolled in comparable sections of the course without the BRIDGES intervention. This shows that the students perceived the value of the assignments, which support the idea that the intervention was beneficial.

6 Limitations and conclusions

However, since the data collected were from field-based studies with a quasi-experimental design, there are some limitations that prevent us from making generalizations about what some of the immediate and long-term performance gains resulted from. Although all the students were enrolled in the same course at the same institution and a common knowledge test was used to assess short-term learning gains, the instructors were different and variation in their grading standards may have played an important role in the student outcomes: especially, since the comparison groups were comprised of multiple sections, each with their own instructor. Also, group sizes were not comparable, with many more students in the comparison group than in the BRIDGES group. Both of these factors would preclude any additional statements about the reasons for the group differences in outcomes. It is also possible that the increased difficulty of the assignments and increased time spent on assignments may help to explain the better outcomes for BRIDGES students.

Another important consideration is that when BRIDGES was used in the data structure courses, the focus had been on the software aspects of implementing data structures. An equally important aspect of this course is in understanding the computational complexity of data structures and their underlying algorithms, as well as the knowledge needed to choose a data structure for a given problem. Early versions of BRIDGES did not address this part of a normal data structures course. Thus, the knowledge gains of students from using BRIDGES also relied on coverage of these theoretical concepts by the instructors and were assessed separately. Over the past two years, we have extended BRIDGES to topics typically covered in courses on algorithm analysis, and emphasizing benchmarking (sorting, graph algorithms on large datasets); this not only promotes the engagement aspects of BRIDGES, but also shows by example, the computational complexity of data structures and algorithms through direct experiments on real-world data. By integrating benchmarking features within BRIDGES, students can focus on algorithm implementation and running experiments on very large datasets, leaving the more tedious work on drawing plots from their implementation outputs to BRIDGES. Additionally, as an aid to instructors, we now maintain a repository of BRIDGES assignments, in a manner similar to other assignment repositories, with learning goals, descriptions, scaffolded code, and expected solutions.

Taken together the evaluation data show that the BRIDGES infrastructure provides a valuable and engaging tool for instruction in some of the foundational computing content associated with a major in computer science. We have demonstrated that BRIDGES provides an engaging experience for sophomore level students in data structures courses, improves their retention of foundational CS knowledge and their progression in the major. Over the past 5 years, we concentrated on using the tool in the sophomore level data structures course. Looking forward, we will extend the development and use of BRIDGES software to assist in the preparation level of incoming students. In large CS programs, students can enter the program with very diverse computational skills and/or preparation levels. For instance, our own program has a significant number of transfer students (with varying preparation levels) entering the program at the sophomore level from nearby community colleges, and instructors face significant challenges in adapting the curriculum to serve the needs of all students. With the addition of a BRIDGES Game API as well as adapting many of the Nifty assignments (that focus on freshmen CS), BRIDGES now supports CS1 and CS2 learning goals and topics (conditionals, loops, simple data structures) across all three programming languages. This will be of benefit to those students who can use the existing assignments from earlier courses as a means to ‘catch up’.

Work on developing and extending BRIDGES is moving forward on several fronts. We are building a repository of BRIDGES assignments (BRIDGES Development Team, 2022); this provides a resource for instructors looking for interesting BRIDGES based assignments, with detailed descriptions, scaffolded code, expected output and solutions (provided on request to instructors). This also helps minimize the time instructors using BRIDGES spend in building assignments for their courses. Secondly, we continue to incorporate new datasets with appropriate assignments. Two examples of these are (1) OpenStreetMap (OpenStreetMap, 2020), which can be used in graph assignments (shortest path, BFS, DFS algorithms) and spatial query based assignments (Quadrees, K-D Trees), and, (2) building an Audio API, that lets users choose and process songs from an external source, for instance, the Genius API (Genius API, 2023). Finally, we are looking into extending BRIDGES into upper division courses such as Operating Systems and Databases. We believe BRIDGES can play a role in building modules that can help reinforce complex computational concepts and their underlying algorithms.

A project assignment survey

The assignment survey consists of a four short answer questions and nine five point Likert (Agreement) Scale questions as follows:

1. What is the essential concept that was learned by completing the assignment? [Short Answer]
2. Why is it important? [Short Answer]
3. How does the concept contribute to understanding how to program? [Short Answer]

4. Rate the level of Java/C++/Python programming experience required by the assignment relative to your experience [more experience than I have, about right for my experience level, less experience than I have]
5. Please rate the amount of experience that you have had in Java/C++/Python programming [none, a little, moderate]
6. Indicate the total number of hours that you spent completing the assignment [0-3,4-6,7-9, 10 or more]
7. While working on the assignment, rate the degree to which you relied on each of the following sources for help. [multiple sources specified]
8. For the question above, if you indicated that you were getting help from other internet site(s) or other source, please write in that site or other source. [Short answer]
9. I am comfortable talking with and seeking help from the teaching assistant. [5-point Likert]
10. I got enough help from the teaching assistant. [5-point Likert]
11. The assignment was relevant to my career goals. [5-point Likert]
12. The assignment was trivial and not essential to learning about computing. [5-point Likert]
13. The assignment increased my interest in computing. [5-point Likert]

Acknowledgements This material is based upon work supported by the National Science Foundation under Grant Nos. 1245841, 1726809, 2142381

Data Availability The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request

References

- AAAI (2018). Model AI Assignments. *Model AI Assignments*. <http://modelai.gettysburg.edu/>
- Baecker, R. (1998). Sorting out sorting: A case study of software visualization for teaching computer science Sorting out sorting: A case study of software visualization for teaching computer science. *Software visualization: Programming as a multimedia experience, 1*, 369–381.
- Bart, A.C., Tilevich, E., Hall, S., Allevato, T. & Shaffer, C.A. (2014). Transforming Introductory Computer Science Projects via Real-time Web Data Transforming introductory computer science projects via real-time web data. *Proceedings of the 45th ACM Technical Symposium on Computer Science Education Proceedings of the 45th acm technical symposium on computer science education* pp. 289–294.
- Beckman, A., McQuaigue, M., Goncharow, A., Burlinson, D., Subramanian, K., Saule, E. & Payton, J. (2020). Engaging Early Programming Students with Modern Assignments Using BRIDGES Engaging early programming students with modern assignments using bridges. *Proc. ccsc central plains*. Journal of Computer Sciences in Colleges.
- Blumenfeld, P., Kempler, T. & Krajcik, J. (2006). Motivation and cognitive engagement in learning environments. Sawyer, R. (Ed.), *The Cambridge Handbook of Learning Sciences* (p. 475–488). Cambridge, MA: Cambridge University Press.
- Blumenfeld, P. C., Soloway, E., Marx, R., Krajcik, J., Guzdial, M., & Palincsar, A. (1991). Motivating project-based learning: Sustaining the doing, supporting the learning. *Educational Psychologist, 26*(3–4), 369–398.
- Bostock, M., Ogievetsky, V., & Heer, J. (2011). D3 data-driven documents. *IEEE Transactions on Visualization and Computer Graphics, 17*(12), 2301–2309.
- Bransford, J. D., Brown, A. L., & Cocking, R.R. (1999). *How people learn: Brain, mind, experience and school*. Washington, DC: National Academy Press.


- BRIDGES Development Team (2022). BRIDGES Assignment Repository. <http://bridgesuncc.github.io/newassignments.html>
- Brown, M., & Sedgewick, R. (1984). A system for algorithm animation. *Proceedings of the 11th annual conference on computer graphics and interactive techniques SIGGRAPH '84*, 18(3), 177–186.
- Buckley, M., Nordlinger, J., & Subramanian, D. (2008). Socially relevant computing. *Proceedings of the 39th SIGCSE technical symposium on computer science education* (pp. 347–351).
- Burlinson, D., Mehedint, M., Grafer, C., Subramanian, K., Payton, J., Goolkasian, P., & Kosara, R. (2016). Bridges: A system to enable creation of engaging data structures assignments with real-world data and visualizations. *Proceedings of ACM SIGCSE 2016* (pp. 18–23).
- Cohoon, J. (2005). *Just get over it or just get on with it. Women and information technology: Research on under-representation*. Cambridge, MA: MIT Press.
- Computing Research Association (2018). CRA Taulbee Survey, 2018–2019. <https://cra.org/wp-content/uploads/2019/05/2018-Taulbee-Survey.pdf>
- Computing Research Association (2019). CRA Taulbee Survey, 2019–2020. <https://cra.org/wp-content/uploads/2020/05/2019-Taulbee-Survey.pdf>
- Dann, W. P., Cooper, S., & Pausch, R. (2005). *Learning to Program with Alice*. Prentice Hall.
- Drake, P., & Sung, K. (2011). Teaching introductory programming with popular board games. *Proceedings of ACM SIGCSE 2011* (pp. 619–624).
- Genius API (Accessed Jan 2020). <https://docs.genius.com/#/getting-started-h1>
- Groovy Graphics Assignments (Accessed July 2019). <https://blog.siggraph.org/tag/groovy-graphics-assignments/>
- Guzdial, M. (2003). A media computation course for non-majors. *Proceedings of the iticse 2003* (pp. 104–108).
- Guzdial, M., & Ericson, B. (2016). *Introduction to Computing and Programming in Python, A Multimedia Approach* (4th ed.). Pearson.
- Havill, J. (2021). *Discovering Computer Science: Interdisciplinary Problems, Principles, and Python Programming* (2nd ed.). Boca Raton, Florida: CRC Press.
- Horton, D., Craig, M., Campbell, J., Gries, P., & Zingaró, D. (2014). Comparing outcomes in inverted and traditional CS1. *Proceedings of the ITICSE 2014* (pp. 261–266).
- Hu, H.H., & Kussmaul, C. (2021). Improving online collaborative learning with pogil practices. *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education* (p. 1330). New York, NY, USA: Association for Computing Machinery.
- Jones, B. (2009). Motivating Students to Engage in Learning: The MUSIC Model of Academic Motivation. *International Journal of Teaching and Learning in Higher Education*, 21(2), 272–285.
- Kussmaul, C. (2012). Process oriented guided inquiry learning (pogil) for computer science. *Proceedings of the 43rd acm technical symposium on computer science education* (pp. 373–378). New York, NY, USA: Association for Computing Machinery.
- Latulipe, C., Long, N.B., & Seminario, C.E. (2015). Structuring flipped classes with lightweight teams and gamification. *Proceedings of the acm sigcse 2015* (pp. 392–397).
- Layman, L., Williams, L., & Slaten, K. (2007). Note to self: Make assignments meaningful. *Proceedings of the ACM SIGCSE* (pp. 459–463).
- Monge, A., Quinn, B.A., & Fadjo, C.L. (2015). Engagecsedu: CS1 and CS2 materials for engaging and retaining undergraduate cs students. *Proceedings of ACM SIGCSE* (pp. 271–271). Retrieved from <https://www.engagecsedu.org/>
- Moskal, B., Lurie, D., Cooper, S. (2004). Evaluating the effectiveness of a new instructional approach. *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education* (pp. 75–79). NCWIT (2018). <https://www.engage-csedu.org/>
- NSF/IEEE-TCPP Curriculum Initiative (2018–2022). Peachy parallel assignments. <https://grid.cs.gsu.edu/tcpp/curriculum/?q=peachy>.
- OpenStreetMap (2020). Openstreetmap. Retrieved from <https://openstreetmap.org/>
- Parlante, N. (2018). Nifty assignments. Retrieved from <http://nifty.stanford.edu/>
- Parlante, N. (2021). Nifty assignments review guidelines. Retrieved from <https://sigcse2020.sigcse.org/reviewers/nifty-review-guidelines.html>
- Pierson, W.C., & Rodger, S.H. (1998). Web-based animation of data structures using JAWAA. *ACM SIGCSE bulletin* (vol. 30, pp. 267–271).
- Pirker, J., Riffnaller-Schiefer, M., & Gütl, C. (2014). Motivational active learning: Engaging university students in computer science education. *Proceedings of ITICSE* (pp. 297–302).

- Sedgwick, R., & Wayne, K. (2020). Case Study: Small World Phenomenon. <https://introc.cs.princeton.edu/java/45graph/>
- Resnick, M., Maloney, J., Monroy-Hernandez, A., Rusk, N., Eastmond, E., Brennan, K., & Kafai, Y. (2009). Scratch: Programming for All. *Communications of the ACM*, 52(11), 60–67.
- Sedgwick, R., & Wayne, K. (2017). Introduction to programming in java. Retrieved from (A Case Study: Small World Phenomenon <http://introc.cs.princeton.edu/java/home/>)
- Shaffer, C. (2011a). Data structures and algorithm analysis in c++. doverpublications.com: Dover Publications.
- Shaffer, C. (2011b). Data structures and algorithm analysis in java. doverpublications.com: Dover Publications.
- SHEF (2019). State Higher Education Finance (SHEF) Report. (Available:<https://shef.sheeo.org/report/>)
- Strahler, J., McQuaigue, M., Goncharow, A., Burlinson, D., Subramanian, K., Saule, E., & Payton, J. (2020). Real-world assignments at scale to reinforce the importance of algorithms and complexity. Proceedings of ccsc north east. *Journal of Computer Sciences in Colleges*.
- Stroebe, W. (2016). Why good teaching evaluations may reward bad teaching: On grade inflation and other unintended consequences of student evaluations. *Perspectives on Psychological Science*, 11(6), 800–816.
- Sung, K., Rosenberg, R., Panitz, M., & Anderson, R. (2008). Assessing gamethemed programming assignments for CS1/2 courses. *Proceedings of GDCSE* (pp. 51–55) New York, NY, USA: ACM.
- VanDeGrift, T. (2017). POGIL activities in data structures: What do students value? *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (p. 597–602). New York, NY, USA: Association for Computing Machinery.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.

Authors and Affiliations

David Burlinson¹ · Matthew McQuaigue¹ · Alec Goncharow¹ ·
Kalpathi Subramanian^{1,2}  · Erik Saule¹ · Jamie Payton² · Paula Goolkasian³

David Burlinson
burlinsond2@gmail.com

Matthew McQuaigue
mmcquaig@charlotte.edu

Alec Goncharow
agoncha1@charlotte.edu

Erik Saule
esaule@charlotte.edu

Jamie Payton
payton@temple.edu

Paula Goolkasian
pagoolka@charlotte.edu

- ¹ Computer Science, UNC Charlotte, Charlotte, NC, USA
- ² Computer Science, Temple University, Philadelphia, PA, USA
- ³ Psychological Sciences, UNC Charlotte, Charlotte, NC, USA