# TUES: Transforming Introductory Computer Science Projects via Real-Time Web Data

Modern day introductory CS students are sophisticated consumers of real-time Web-based data; they find a world without e-mail, instant messaging, smart phones, etc. hard to imagine. And yet programming projects assigned in introductory courses commonly have little to do with students' perceptions of what computing is about and why they became interested in it. Finding their introductory classes disconnected from their day-to-day computing experiences, CS students may feel discouraged from pursuing computing as a major. Further, to meet the demands of the computing industry for more qualified and diverse graduates, students need better training in distributed systems. We seek to address these concerns through a framework and supporting API for accessing real-time Web-based data. These will allow instructors of introductory programming courses to give programming assignments that involve the use of such real-time data derived from distributed systems.

To better connect introductory CS curricula with the students' computing experiences, and the realities of modern programming, this proposal will focus on the cornerstone of introductory curriculum—the programming project. We propose a real-time Web data framework that will enable a new generation of programming projects that can better engage and better train introductory CS students. Specifically, we propose to create an educational infrastructure to support programming projects that use real-time Web-based data. Examples of real-time data accessible over the Web include stock values, traffic information, weather, public transit, flight information, inventory overviews, and many others. Our educational infrastructure will make it possible to assign programming projects that use such real-time data on the Web. The infrastructure will play the vital role of ensuring the requisite data availability, integrity, and accessibility, as well as automating grading techniques, vital for classes with large enrollments.

We will begin by improving the infrastructure support for our highly successful BusTracer.EDU API, which is already being used in the CS2 course at Virginia Tech. The existing infrastructure will be broadened to make it reasonably easy for instructors to provide access to other distributed data providers. We will prove our concepts by creating bindings to our framework to support a second data source, such as Yahoo! Finance.

To validate the effectiveness of our educational infrastructure, we will introduce new programming projects for the traditional CS2 course, and also the data structures and algorithms course (both typically taken during the sophomore or junior years). We will evaluate the programming projects for their effectiveness in improving educational outcomes in the introductory classes. We will study their effect on retention rates and on learning relevant skills and content. We will make all created infrastructure and programming projects publicly available on the Internet for adoption and experimentation by peers in other institutions.

**Intellectual Merit:** By bringing real-world, real-time data of intrinsic interest to programming projects assigned in introductory classes, we will not only increase retention across the board but will also introduce students to important advanced concepts of modern software construction (e.g., distributed computing, cyber-physical systems, etc.). The proposed work presents an opportunity to fundamentally reconsider whether CS education adequately prepares our students for the emerging challenges of the IT workplace. Making introductory programming projects more relevant and engaging will positively impact all aspects of these introductory courses. This work will contextualize programming projects in introductory programming classes using real-time data provided by real-world Web-based systems. It aims at introducing some of the most exciting computing technologies to introductory students. Thus, our goal is both to increase student engagement and to expand the content that students traditionally learn in introductory courses.

**Broader Impacts:** The proposed project aims to increase recruitment and retention of students in the Computer Science major. It has particular potential to increase participation by women. Prior research indicates that women are more likely than men to study computer science as an enabling technology for meaningful pursuits in non-computer science domains. Thus, appropriately contextualizing programming projects is likely to render CS more relevant and attractive to a larger percentage of introductory students.

## 1 Problem Statement

Although undergraduate students interact with distributed applications (e.g., e-mail, social networking, Web, mobile devices, etc.) in their day-to-day lives, they are typically introduced to distributed system topics only in advanced classes, if at all. CS curricula have not kept up with the dominant computing system trend—with computing becoming increasingly distributed. Not covering any distributed system issues in introductory classes contributes to the perception of CS as a discipline that is disconnected from the real world, making it hard for students to contextualize their interest in CS within a larger purpose. In fact, prior research suggests that competent students leave CS discouraged by the lack of immediate applicability or relevance for what they have learned [4, 9]. Gender-specific differences play a crucial role in determining what drives students to CS [6]: using computing as a tool in another field is a primary motivation for women, while for men this motivation ranks only third [4]. As an academic discipline, CS is plagued by high attrition rates, which average between 30% and 40% during the freshmen and sophomore years [2].

Computing educators have been continuously looking for better alternatives to using canned programming assignments from mainstream textbooks in order to increase the appeal of the discipline for a broader and more diverse student audience [1, 3, 6, 7, 11, 12, 16, 17, 23]. The recurring themes shared by these efforts include using real-world problems relevant to students' lives, highlighting the social and societal impact of such problems, creating solutions whose utility extends beyond the classroom, and empowering students with flexibility to creatively control their solutions.

Since the computing systems which which students interact on a regular basis are increasingly distributed, the thesis of this proposal is that introducing distributed computing into programming projects early in the curriculum has great potential benefit. This early exposure to distributed computing can help close an important gap in student knowledge, and to raise their level of interest due to working on relevant problems.

## 2 Solution Overview

Compared to traditional engineering disciplines, Computer Science is particularly vulnerable to teaching styles that lack contextualization, particularly in introductory courses. To demonstrate this point, consider the typical experiences of students starting to study Mechanical Engineering and CS. Introductory classes in both disciplines are commonly project based. However, the actual projects assigned in each respective discipline are likely to differ drastically. While Mechanical Engineering course projects often deal with real world objects (e.g., assembling a mechanical arm), CS assignments are typically concerned with abstract data manipulation (e.g., sorting integer values).

While the ability to manage abstraction is one of the cornerstones of CS as a discipline, that is different from 'abstract' in the sense of no contextualization. Relying solely on abstract concepts in introductory classes can be discouraging for some students. When comparing their experiences to that of their peers who are "building real stuff" (that occurs in Mechanical Engineering classes), some CS students perceive their courses as lacking any real world relevance and may lose interest in pursing CS as a major. This is one problem that this proposal aims to address.

To concretely demonstrate the issues outlined above and how the proposed solution will be applied, consider VT Bus Tracker (`www.bustracker.org`), an information system that enables passengers of Blacksburg's local transportation system to track the location and movement of commuter buses over the Internet and using mobile devices. Bus Tracker represents a Web-based information system that is immediately relevant to Virginia Tech students. Bus Tracker started as a project in a capstone course taught by PI Tilevich; then the Virginia Tech Student Government Association (SGA) funded Tilevich to lead a team of students to release the system for real-world use beginning in the 2010-2011 academic year. If other academic institutions could leverage their local systems as suppliers of real-time Web-based data, such data would be perceived of intrinsic relevance by their students.

Assume that the educational infrastructure that we propose has been applied to VT Bus Tracker, and a convenient Application Programming Interface (API) has been provided for educational purposes. The API will expose the entire functionality of the system through a set of classes in a mainstream Object-Oriented programming language such as Java or Python.

To demonstrate the potential of our proposed framework to improve the relevance of programming projects, consider two versions of the same basic programming assignment. They exercise essentially the same programming skills, and either can be assigned in an introductory programming class.

**Original Assignment:** *Read a list of data records from a disk file and add them to a doubly linked list in their order of appearance. Remove all the records whose field X is greater than 10. Print the lists before and after the removal.*

**Restructured Assignment:** *Read a list of buses on Route X using the provided BusTracker.EDU API and add them to a doubly linked list in their order of appearance. Remove all the buses whose passenger count is greater than 20, as these buses are unlikely to be able to pick up any more passengers. Print the pruned list of buses in the order of arrival to a stop of your choosing.*

While both assignments cover the same CS content and skills (i.e., manipulating a doubly linked list), the restructured assignment is relevant to the day-to-day lives of the students, as it resembles the type of route-planning applications that they already use. Instead of manipulating abstract objects in what seems a purely theoretical undertaking, the restructured project places the assignment in a real world context. By using real-time distributed data, students have to appropriately deal with the resulting variabilities. Not only do students interact with data representing real objects (i.e., commuter buses), but the final deliverable of their project can be further developed into a tangible information asset with the potential to concretely benefit students. Entrepreneurial students can even choose to follow up on this assignment in natural ways, developing a mobile application that can be distributed to others. Finally, with a suitable interface, reading real-world Internet data is no more complex than reading data from a disk file.

But Bus Tracker is just one example of real-time data from a distributed system. We seek more broadly to supply a framework, interfaced through an API, that can be tailored by faculty at other institutions to interact with a broad range of distributed systems.

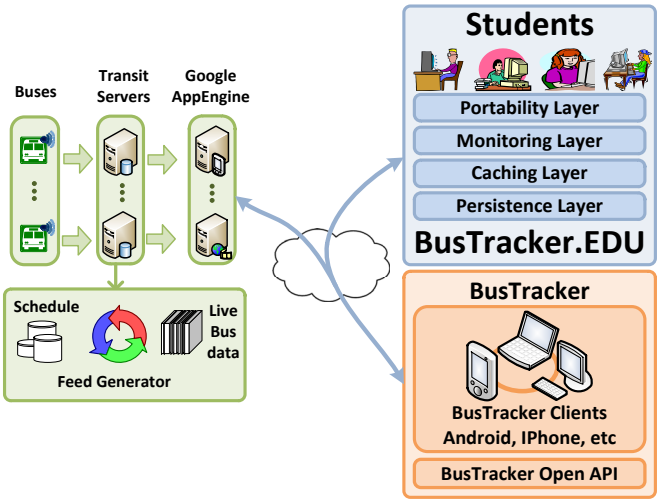


**Figure 1. Example infrastructure instantiation.**

Figure 1 shows how the proposed educational infrastructure can furnish real-time Web-based data for student projects. The infrastructure's instantiations (such as our own BusTracker.EDU) will provide persistence, caching, monitoring, and portability capabilities to allow students to work on programming projects with readily available and reproducible distributed data. Students should be able to test their assignments on specially provided data sets or to specify the time period during which they tested their deliverables. Graders should then be able to emulate the specified time periods from a historical archive.

The purpose of the proposed infrastructure is to make a broad range of real-time Web-based data available for educational purposes with only modest effort by local faculty. Distributed Web-based applications serving real-time data have become ubiquitous due to the proliferation of mobile devices such as smart phones. Stock information, arrival/departure notifications, news updates, and price fluctuations

have all been exposed through distributed applications that furnish these real-time data to the user. The goal of this proposal is to allow faculty to leverage these data to revitalize programming projects, particularly in introductory CS classes.

To ease transfer and adoption, the infrastructure will have an extensible architecture realized as an open-source solution. As such, the infrastructure will be applicable to a variety of CS courses. One type of programming project will involve scenarios driven by real-time data provided by various Web based systems relevant to specific educational institutions. Another type of programming project involves analyzing historical data of these systems that the framework will make possible through storage in a database. Both the real-time data and the historical data can be used to motivate programming assignments and exploratory projects for a variety of CS courses and independent study projects, both undergraduate and graduate. It is also possible to devise a transfer pathway for students to "publish" the best of these projects as mobile or web applications available to the broader community, thus creating a self-sustaining educational software ecosystem.

The proposed infrastructure will provide an avenue for introducing important CS concepts early in the curriculum. Computing is at a crossroads, and the next several years will see a fundamental shift in how the average user takes advantage of computing resources. Traditional desktop software applications will move in the direction of a computation model dominated by cloud computing. The proliferation and popularity of mobile devices will lead to the distributed programming model becoming primary. It is never too early to introduce students to the challenge of engineering and interacting with distributed systems. Thus, the new programming projects based on the proposed educational infrastructure will expose students to distributed computing at the appropriate level, as the advanced challenges of distributed computing (i.e., coordination of multiple distributed resources, failure detection and handling, latency, etc.) will be handled by the infrastructure.

Thus one goal of the proposed research plan is to explore whether programming projects placed in a real-world context can improve retention of CS students. Another goal is to investigate how advanced distributed computing concepts can be introduced early on to the standard CS curriculum, thereby better preparing students for the realities of constructing modern software.

## 3 Accessing Web interfaces through APIs

A Web application programming interface (API) is a set of methods for communicating with a Web application to request and submit data from a client application. Web APIs enable richer, user-oriented, and customized applications. Multiple API's can also be combined into a mashup. Web APIs have become ubiquitous and enable various non-browser interfaces to Web applications, such as native desktop and smart phone clients. The majority of popular Web applications provide APIs, including Google, Facebook, Flickr, and YouTube. Multiple existing API examples can be found at `http://www.programmableweb.com`.

Rapidly becoming an industry standard, Web APIs define methods that adhere to the "RESTful" software architecture style. REpresentational State Transfer (REST) relies on four actions: GET—request data, PUT—replace data with new data, POST—create new data, and DELETE—erase data.

REST-based APIs work through HTTP, just like a website. For example, to download some Flickr photos, the following method from the API can be invoked:

```
http://flickr.com/services/rest/?method=flickr.photos.search&
api_key={API_KEY}&text=Virginia%20Tech&per_page=50.
```

In response, Flickr returns an XML document. Other common formats returned by API methods include JSON and HTML. An object-oriented language can provide a wrapper class that encapsulates all the complexities associated with interacting with a Web application through a raw REST-based API. Consider using a Flickr API in an application. All the API methods can be defined in an object called flickrApi .

- flickrApi .getPhotos( Virginia Tech ); will search for all the photos related to Virginia Tech,
- flickrApi .putNewCaption(Some **new** caption, photo.id); will change a photo's caption,
- flickrApi .postNewPhoto(newPhoto, New photo caption) will upload a new photo, and
- flickrApi .deletePhoto(photo.id) will delete a photo.

Notice how these methods correspond to the get, put, post, and delete verbs of the "RESTful" architectural style.

There are several technological challenges to using Web data accessible through the APIs directly in introductory programming projects.

- Creating Object-Oriented wrappers that are intuitive enough for introductory students requires a high level of sophistication in software design and implementation.
- Obtaining and managing API keys incurs an administrative overhead. Web application companies may have different policies regarding their API keys depending on how much users plan to access their application through the provided API. It may be free under a certain threshold, and then a usage fee may be incurred.
- Web APIs returning real-time data are not idempotent. The same method invoked multiple times with the same arguments are likely to return different results. This lack of idempotency significantly complicates testing and grading.
- Web data might not be constantly available. For example, a Web application providing transit arrival information might not provide data during off hours.

The major product of this proposal will be a software framework that will make it possible to use real-time Web-based data for educational purposes. Our infrastructure will address the technological impediments described above.

A rule of thumb in system design is that one cannot successfully engineer a general system for multiple cases until one has built a concrete system for a specific case. In that light, the proposed work will generalize our experience from successfully leveraging VT Bus Tracker for educational purposes. We next describe this experience and the lessons we have learned in the process.

## 4 Prior Work: The BusTracker.edu Educational Infrastructure

Blacksburg Transit (BT) operates several routes of commuter buses in the town of Blacksburg (where Virginia Tech is located) and its environs. For more than five years, BT had been collecting real-time arrival data for their buses by means of GPS receivers installed on each bus. Although BT had always archived the information collected by the receivers, it had used the information exclusively for scheduling and reporting purposes. BT partners with the Google Transit program, which integrates public transportation information with the planning functionality of Google Maps. Through this partnership, BT started using Google's Transit Feed Specification (GTFS) [10] to periodically transmit to Google the static information about their bus routes, stops, and schedules. This static information, however, does not include any real-time bus location and movement information, nor can GTFS be accessed through a client side Object-Oriented language API.

In the spring 2009 semester, PI Tilevich taught a Software Engineering senior capstone course that enables students to apply their CS knowledge and skills to solve a real-world problem. Instructors of our capstone courses often choose to partner with not-for-profit organizations that need to solve a problem using IT but have insufficient in-house resources to do so. Such a partnership is a win-win arrangement: it allows the students in the course to validate their skills by creating an IT solution to a real world problem, and it gives the organization a functional IT solution at no cost.

PI Tilevich chose to partner with BT, which for some time wanted to go a step beyond their existing partnership with Google by making productive use of their real time bus information as well. The capstone

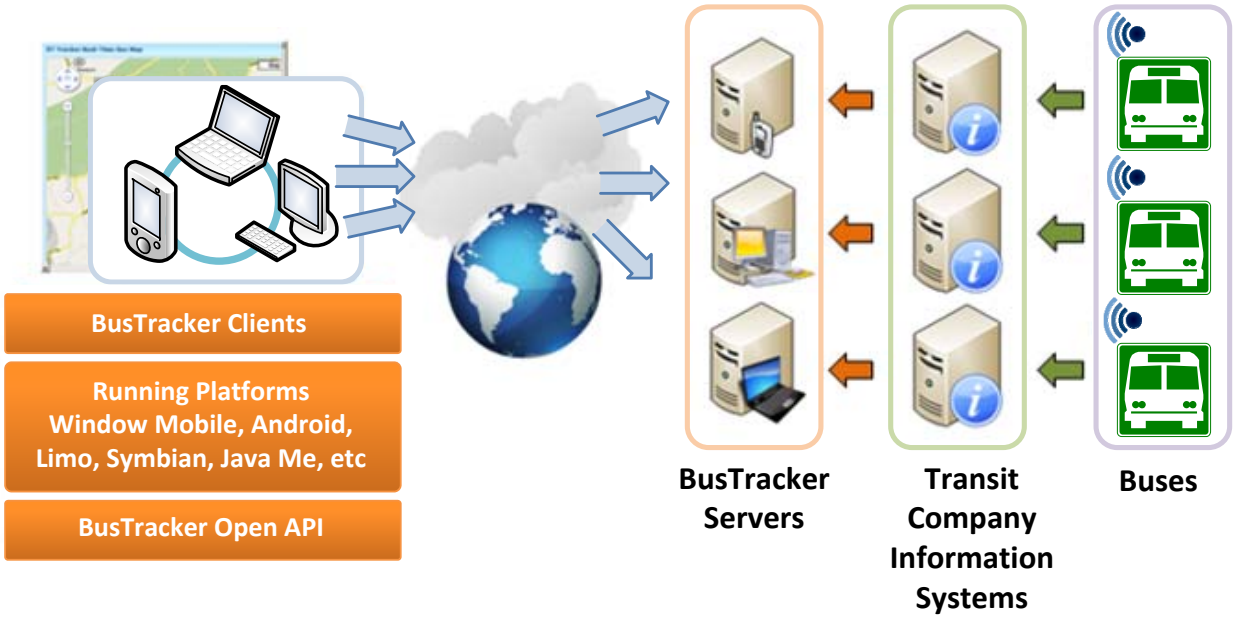


**Figure 2. The VT Bus Tracker Main Architecture.**

course presented a perfect opportunity for this undertaking. Over the course of a single semester, ten senior-level students were able to render the real-time BT data accessible to the passengers. Under Tilevich's supervision and in collaboration with BT, the students developed BusTracker, a working information system prototype that enabled the user to access BT's real-time bus arrival and movement information through a Web browser.

BusTracker is a cyber-physical system, that is, a system that tightly integrates physical and computational elements. GPS receivers reporting real time bus whereabouts constitute the physical element. This information is then relayed to passengers by means of a sophisticated information system. The physical and computational elements of BusTracker work in concert to provide passengers with intelligent transportation information that can inform their bus-riding decisions.

BusTracker has received positive publicity and has been enthusiastically embraced by Virginia Tech students and faculty. Citing transportation accessibility as one of its primary concerns, the Virginia Tech Student Government Association (SGA) chose to fund the project over the course of Summer 2010. The SGA grant enabled PI Tilevich to lead a team of students to mature and enhance the system, which renamed to VT Bus Tracker was officially released by the beginning of the 2010-2011 academic year.

Figure 2 depicts the VT Bus Tracker architecture. The architecture has four main layers. First, commuter buses are equipped with GPS units which periodically transmit (using a proprietary radio protocol) their whereabouts to the Blacksburg Transit information system. Bus Tracker pushes the real-time bus information, augmented with static data regarding the schedule in effect, to the Bus Tracker server, hosted by the Google App Engine (GAP [5]), a platform to program and host Web applications on the Google cloud infrastructure. Using a cloud infrastructure allows cost effective scaling of the provided service on demand. To access the information provided by Bus Tracker servers, we have designed and make available a Java API that constitutes the backbone of our development framework. Bus Tracker clients use this API to retrieve information pertaining to the location and movement of BT buses.

Figure 3 provides a more detailed overview of the Bus Tracker information system. The diagram in the middle depicts interactions between the BT company servers and the Bus Tracker feed servers. Bus data

are both static and dynamic. Static data includes the schedule, which is usually decided at the beginning of the semester and changes infrequently. Following the standardization trend of the transportation industry, the schedule is provided in Google's General Transit Feed Specification (GTFS) format [10]. Dynamic data encompasses all the updates received from GPS devices located on buses that are in service. The static and dynamic information is reconciled into a single feed that uses JavaScript Object Notation (JSON). The reconciled feed, hosted by Google App Engine, can be accessed by any client through HTTP. To facilitate processing of feed data, we provide the Bus Tracker API, a set of Java classes that represent various feed objects (e.g., buses, routes, schedule, etc.) along with methods for accessing their data. Bus Tracker clients may choose to forgo the use of our API and access the feed directly, but then they would have to implement functionality to retrieve and parse the feed. Furthermore, if the feed format were to change, their implementation would have to be modified accordingly. By contrast, using our API shields clients from changes in the feed's format since our development team, in coordination with BT, would change the API bindings in a timely manner. A typical Bus Tracker client is a mobile device, such as a smart phone, that uses the API to retrieve and display relevant information in a GUI. Bus Tracker does not support any GUI rendering, as GUI frameworks are usually specific to each mobile device.

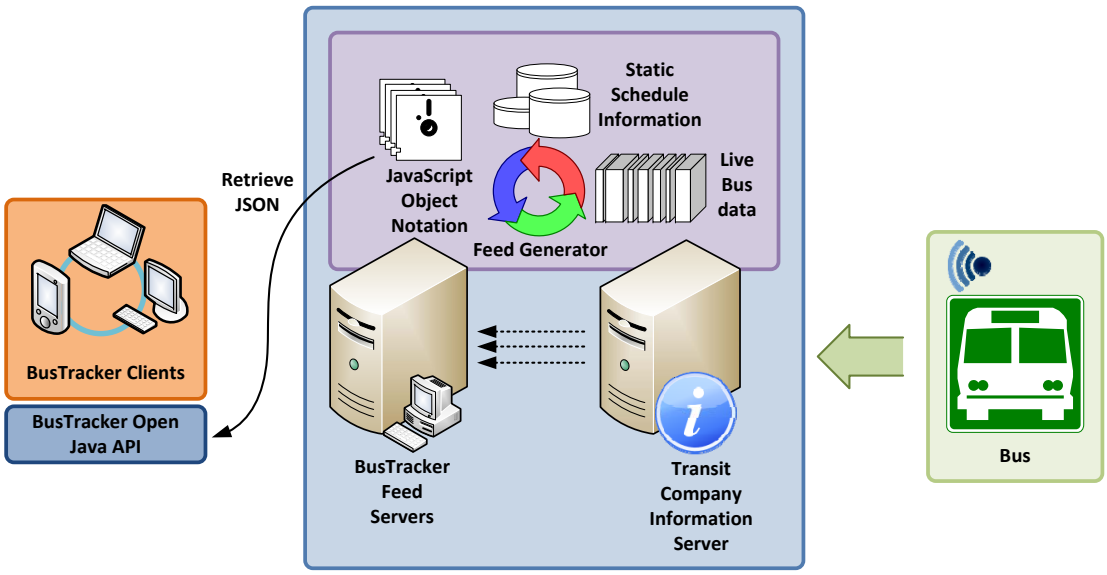


**Figure 3. The BusTracker System Overview.**

Once Bus Tracker was released to BT passengers and received good publicity, we naturally wanted to leverage the system for educational purposes. To that end, we created an experimental prototype—BusTracker.EDU—and used it to design and teach a special laboratory taught in a CS2 ("Data Structures and Software Design") class taught by PI Tilevich.

The prototype infrastructure includes a middleware system and the Application Programming Interface (API) for accessing the system. The BusTracker.EDU API extends the BusTracker API described above. The added functionality makes it possible to design programming projects.

Written in Java, the API provides a set of classes that will be used in student projects. The methods of these classes can be invoked to request structured bus data from BusTracker.EDU.

We next summarize the system requirements for building a robust and usable educational infrastructure.

## 5 Educational Infrastructure System Requirements

Based on our experiences of leveraging VT Bus Tracker real-time data for student projects, we discovered a set of major requirements that a general infrastructure for educational purposes should fulfill. We present these requirements in this section, and explain how we satisfy them in Section 6. It is possible that working with other Web real-time sources will reveal additional minor requirements, but the requirements described next present a solid starting point of the proposed investigation.

**Ensuring the requisite data availability, integrity, and accessibility:** To leverage the data provided by a real-world Web-based system in educational contexts requires that certain guarantees be provided. Users of the proposed educational infrastructure are not only non-experts, but people who are only starting to acquire the necessary skills and confidence in the discipline of Computer Science. If the envisioned infrastructure is to provide the expected educational benefits, it should not unduly frustrate the students by imposing additional work that would not be conducive to the overall educational experience.

To that end, the data provided must be readily available for use by both the base system and the educational infrastructure. The availability of Web data depends strongly on the time of day. For example, a transit company may offer limited night-time service, reporting few real-time scheduling updates. Nevertheless, students may choose to work on their programming projects during off-peak hours. To accommodate the diverse working preferences of different students, they should be able to run traces from the historical data archive as though it were being received in 'real time'.

The provided data should be verifiably correct, at least with respect to its schema. For example, data provided by a cyber-physical system can become corrupted due to physical environmental factors. There could be errors when acquiring data via sensors, transmitting it between different systems, and retrieving it for individual projects. A failure in one part of the cyber-physical system can affect all the other parts of the system that depend on it. As a specific example concerning Bus Tracker, an incorrect sensor reading of passenger count propagates erroneous assumptions to the rest of the system. A system component using these passenger counts to direct buses that have spare capacity often suffers in turn.

To be a viable educational resource, the provided data must be readily accessible at all times. By accessibility, we mean both the ability to retrieve the data through an intuitive API and not being susceptible to service interruptions from system failure or malicious interference.

**Ensuring API usability:** As mentioned above, any information from the system should be accessible through a Java API. The API will be read only, in the sense that data can only be retrieved, so that its copy can be modified locally if necessary. Students should not be able to change any shared information. The API should be able to abstract the structural content of the feed, so that the clients can extract arbitrarily complex data structures using simple method calls operating on an intuitive object model.

We envision that instructors composing the programming assignments that use the API will provide a set of clear instructions to the students regarding how the API should be used. However, for these instructions to be meaningful to the average student, significant care must be taken in designing the API to adhere to well-known Software Engineering principles. For example, following standard naming conventions, ensuring strong encapsulation, supporting unit testing, and documenting the code copiously.

Since many students learn best by example, their learning style must be accommodated by incorporating example applications that exercise the complete set of functionality exposed through the API.

**Accommodating semi-automated grading techniques:** Introductory courses tend to have large enrollments, thus presenting a significant grading burden. It is essential to be able to take advantage of the state-of-the-art automated grading techniques and tools. Among such tools are specially enhanced unit testing harness modules that assign grades by analyzing the number of failed tests and the overall coverage of the tested code base. At Virginia Tech, our colleague Steven Edwards leads a highly successful automated testing initiative called Web-CAT [8].

One of the biggest challenges to using tools such as Web-CAT is ensuring reproducibility. To that end, students should be able to test their assignments on specially provided data sets, rather than actual real-time data. Alternatively, individual students should be able to specify the time period during which they tested their project in order to furnish the required functionality. Graders should then be able to emulate the specified time periods from the historical archive.

## 6 System Architecture

To address the technical challenges explained above, we propose to implement our educational infrastructure using a layered architecture. These layers represent abstract architectural blocks that can be realized through a variety of concrete implementations. In our description for each proposed layer, we first describe the general functionality and then discuss how it can be implemented.

**Persistence Layer:** To provide consistency guarantees for the supplied data, a *persistent layer* will maintain a historical data log, retrievable as though it were being generated in real time. It can be realized as a database-backed subsystem. Using a relational database is an industry proven solution to multiple potential problems of data unavailability and inaccessibility. A standard SQL interface also makes it possible to assign a large range of projects based on analyzing the historical record, for assignments related to data mining, optimization, and planning.

**Caching Layer:** In the presence of network volatility or access bottlenecks, a *caching layer* can provide resilience and fault tolerance. The client APIs will be able to work with such a caching layer. When this option is selected, the API in place will function as if using live data for all its methods. As a result, students will be able to work on their projects and test their functionality in the absence of a reliable network connection. As an additional benefit, caching should significantly reduce the overall load on the feed server. We envision that students will be encouraged to take advantage of the caching layer, unless their project explicitly requires live data. Of course, such applications will be able to easily switch to live data by simply redefining the input data source.

A common approach to implementing caching is to use an RDBMS that caches the required portion of the historical record, making it available locally. The caching layer can emulate the functionality of making actual remote requests to the main data server. The data is cached upon the first access of a particular feed snapshot.

**Monitoring Layer:** To be able to monitor the system's execution to discover execution hotspots and common failure points, a *monitoring layer* can aggregate anonymized usage data, making it available for future analysis. Monitoring addresses the impossibility of predicting how a realistic system will be used over time by a large number of users. For example, if some API methods are discovered to be consistently throwing certain exception types, such methods may need to be redesigned to make their parameters and use more intuitive.

**Portability Layer.** To ensure that our educational infrastructure is easily portable, a *portability layer* will ensure that the infrastructure services can be easily configured to work with different Web applications and operational environments. Our ultimate goal is to enable educators in peer institutions to use our infrastructure to make real-time, Web-based data from other applications relevant to their own institutions available for educational purposes. The portability layer will simplify the adaptation of real time Web data feeds as an input source for our infrastructure. As an implementation strategy, we will offer various adapters and proxies that can transform between new Web data feeds and some existing solutions developed using our framework.

Figure 4 shows the architectural diagram of the proposed infrastructure's instantiation and how it will interact with its underlying Web application. In essence, the educational infrastructure becomes a special-purpose client for its base Web application. Similar to, for example, a mobile client that uses the data to present it to the user in a GUI, the infrastructure will make the data available for educational purposes such

as programming projects. Rather than passively consuming the obtained data, the educational infrastructure adds additional capabilities, including persistence, caching, monitoring, and portability. Another way to think about the infrastructure is as a proxy for the main Web application. Through proxy adaptations, the proxy can flexibly service client requests as required by the situation at hand. For example, if some data from a currently unavailable service is requested, that particular API call can return cached results from an earlier snapshot. The infrastructure will also be configurable to return a fixed set of results during multiple invocations to facilitate testing and debugging.

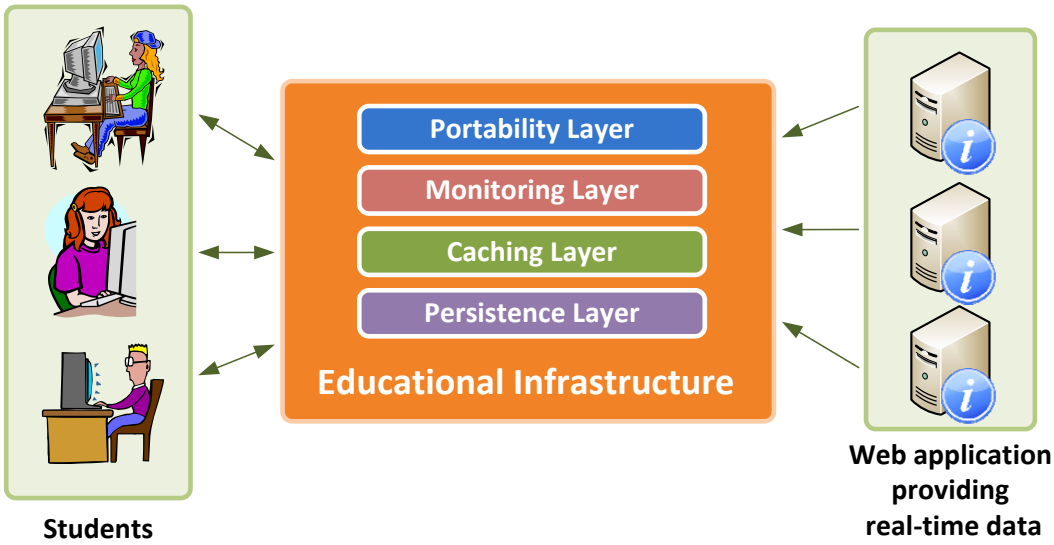


**Figure 4. An Architecture of Proposed Educational Infrastructure.**

## 7 Educational Infrastructure Implementation Plan

We propose to leverage the lessons we have learned developing and evaluating BusTracker.EDU to create a general educational infrastructure for integrating real-time Web-based data into programming projects. Our vision is to build a parameterizable system that can be instantiated with any real-time feed, immediately equipping it with persistence, reproducibility, and caching. It is these properties that are required to be able to use the data for programming projects in introductory classes.

To transform any third-party, real-time, Web-based data into a viable educational resource, the proposed work will apply known Software Engineering principles and technologies. To that end, we will create an extensible *software framework*. Libraries and frameworks are the cornerstones of modern software development—they provide predefined, reusable software building blocks. The difference between libraries and frameworks is subtle but essential. While both libraries and frameworks provide reusable pieces of functionality, frameworks also define an architecture that the developers can follow to implement their applications. A typical Object-Oriented software framework comprises a collection of classes that define some core functionalities. To add application-specific functionality, the developer can then add custom classes. These classes can extend some abstract framework classes or interfaces or be written from scratch to fit certain requirements (e.g., a class should have a no-arguments `public` constructor).

We propose to implement our educational infrastructure as a software framework. CS instructors will be able to use this framework to render real-time Web data usable for educational purposes. The instructors would still have to write some code, but the framework will significantly facilitate this task. The persistence, reproducibility, caching, and portability functionalities will be partially implemented by the framework. The instructors would only have to write the code that customizes these functionalities for their educational

institutions and needs. The instructors will then be able to pick and choose which functionalities they want to include in each programming project.

Take for example using real-time stock information for educational purposes. As a specific example, consider the information provided by a Web site such as Yahoo! Finance (`http://finance.yahoo.com/`). Most market exchanges have well-defined hours of operation, which may prove to be a limiting factor when using stock data for student projects. Some CS students prefer to work on their projects at night when stock exchanges are closed. The proposed framework will make it possible to make stock data available at all times. To that end, course instructors will be able to use our framework to add persistence and caching to the API for accessing stock market data. The instructor will need to specify the amount of data that will need to be cached and persisted as well as its expiration policy. The framework will provide the essential caching and persistence components. As a result, rather than solving this problem in an ad-hoc fashion, the instructors will be able to configure and customize the framework's core classes. The net effect will be that students will be able to use the resulting API to operate on historical stock data, which will also simplify testing and grading.

In addition to building the described software framework, we will also document the best practices of creating APIs for student projects. Certain Software Engineering principles are conducive to creating intuitive APIs that are easy to learn and use. For example, we have observed that date and time information is better specified as a single Date object rather than as a set of multiple parameters. When given two different versions of an API that used these two approaches to specify data and time information, students working on a lab assignment found the latter version more intuitive with fewer date-related bugs in the submitted lab solutions. We will create an on-line catalog of these practices for use by instructors in peer institutions.

## 8 Impacts on the CS Curriculum

In this section, we give reasonable scenarios for using the proposed infrastructure in actual courses. Figure 5 demonstrates the capabilities of the BusTracker.EDU API by implementing a simple program that tracks movement of the first bus on a given route. As can be seen in this code snippet, the API is straightforward to use, and it does not use any special features not commonly found in the kinds of third-party libraries typically used by sophomore and junior CS students. All middleware functionality required for communication and distributed processing is encapsulated inside a JAR archive file that the students working on a new project would simply place into their CLASSPATH.

We introduced new projects using the BusTracker.EDU API within the traditional CS2 course during Spring 2011, with positive initial results. An informal survey confirmed that the vast majority of students found the experience using this Web API interesting and useful. All the forty two voluntary survey participants (out of about a 60 students who participated in the experiment) indicated that they found the experience of using real-time Web data interesting and would welcome more such programming projects in the future.

Another possibility is to use the infrastructure to enrich projects in our sophomore/junior data structures and algorithms course (that we refer to as CS3). At Virginia Tech, these courses are typically taken during the sophomore or junior years. In our CS2 course, students are introduced to the fundamental principles of object oriented programming, some basic data structures (linear structures, binary search tree), algorithms (simple sorts), and GUI programming. The programming projects assigned in CS2 are typically either concerned with building GUIs or using some data structure. We believe that both types of projects can be made more relevant by using Web APIs. The proposed infrastructure can provide relevant and interesting data for building GUIs. Most data structures covered in CS2 (arrays, lists, BST) can be exercised with the new programming projects we envision.

For the third semester data structures and algorithms course, we plan to take advantage of the historical information provided by the system. One topic covered in this course is the representation of graphs and some standard graph algorithms. A big issue in transportation systems is planning. The historical data from

```
1  package BusTrackerExample;
2
3  import java.util.ArrayList;
4  import java.util.Hashtable;
5  import java.util.List;
6  import java.util.Scanner;
7  import org.vt.bustracker.edu.*;
8
9  public class APIDemo
10 {
11
12   public static void main( String[] args ) throws InterruptedException
13   {
14     BusTracker bt = new BusTracker(); //create main object
15     List<Route> routes = bt.getActiveRoutes(); //get a list of active routes
16
17     Scanner sc = new Scanner( System.in );
18     System.out.println (" Enter a route:'' ); //read user input
19     String routeName = sc.nextLine();
20
21     Route route = bt.getRouteByName( routeName ); //get requested route
22     List<Bus> buses = bt.getbusesOnRoute( route ); //get a list of buses
23     trackBus( buses.get(0) ); //pass the first bus on the list
24   }
25
26   public static void trackBus( Bus b ) throws InterruptedException
27   {
28     GlobalPosition last = b.getPos(); // get GPS position
29     GlobalPosition current = b.getPos(); // get GPS position
30     double totalDistance = 0.0;
31
32     while ( true ) {
33       if ( b.isActive() ) { //while bus is moving
34         current = b.getPos(); //keep updating the GPS position
35         if ( ! current.equals( last ) ) { // if the bus has moved
36           totalDistance += current.getDistanceTo( last );
37           System.out.println(current); // print distance travelled
38           System.out.println (" Distance traveled = '' + totalDistance);
39         }
40         last = new GlobalPosition(current); // last GPS pos become current
41
42       }
43       Thread.sleep(5000); //wait for 5 secs
44     } //while
45   }
46 }
```

**Figure 5. A simple program demonstrating the BusTracker.EDU API.**

***Step 1. Infrastructure Design and implementation***
(a) Analysis and Requirements solicitation
(b) System design and implementation
(c) Documentation
(d) Validation and testing

***Step 2. Prog. Projects Design and Implementation***
(a) Create project descriptions
(b) Prototype project implementation

***Step 3. Assign Projects in Class***
(a) Assign in Data Structures class
(b) Assign in CS2 class

***Step 4. Evaluate***
(a) Assess effectiveness via student evals.
(b) Give and evaluate questionnaires

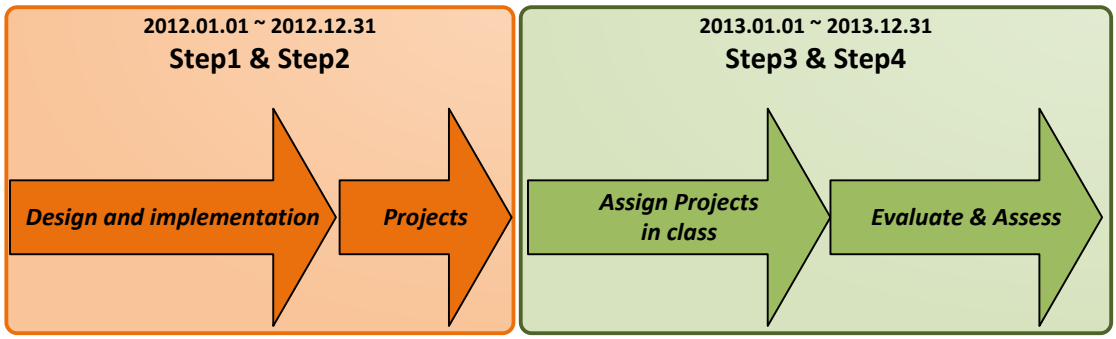


**Figure 6. The proposed work plan.**

the system can be used to populate a graph that can then be used for running planning algorithms. An application might be to suggest optimal itineraries based on given constraints.

## 9 Work Plan

Figure 6 shows the proposed schedule for this project. We plan to spend the first year designing and developing the proposed educational infrastructure by leveraging the lessons we have learned from our experiences with BusTracker.EDU. This work will be supervised by Dr. Tilevich, with Dr. Shaffer testing for usability. Concurrently, we will also design and prototype new programming projects based on the infrastructure. Dr. Tilevich will devise projects for the CS2 course, while Dr. Shaffer will devise projects for the Data Structures and Algorithms (CS3) course. During Spring semester of the second year (2012), we will assign projects using the developed educational infrastructure in selected sections of the CS2 course taught by Dr. Tilevich. Assignments will be refined and repeated in CS2 during Fall 2012. In addition, more sophisticated assignments will be introduced in a section of the CS3 course taught by Dr. Shaffer. Throughout the second year we will assess and summarize our findings regarding the effectiveness of the created research infrastructure and their impacts on the courses and the overall curriculum. We will produce bindings for at least one additional data source unrelated to BusTracker, and use the result in class assignments.

## 10 Evaluation Plan

Following recommendations of Moskal, Lurie, and Cooper [15], we will measure multiple indicators of retention impact of the introduced projects. The first goal for this project is improving retention rates for the relevant courses. Our department routinely collects relevant data both in terms of successful pass rates for the courses in question, and for the point at which students leave the program. By comparing these numbers before and after our intervention, we will determine our approach's impact.

To be successful, the software infrastructure and the projects must satisfy the following requirements: (1) the APIs need to be understandable and easy to use; (2) data access needs to be reliable; and (3) the size and intellectual difficulty of the projects need to be appropriate to the courses. We will survey the students to determine their opinions on these aspects of the infrastructure and the project requirements. We will assess the benefits of exposing students to distributed computing issues early in the curriculum by surveying them

subsequently in later classes to assess whether such early exposure helped prepare them for these classes and extracurricular activities such as summer internships.

More broadly, we seek to improve student appreciation for the real-world applicability of Computer Science. A recent report on student and faculty attitudes [14] is relevant to this issue. Lewis, et al. surveyed on a number of questions related to creativity, self-efficacy, and relevance. We will perform a similar survey on a subset of their questions before and after exposing students to the API-based projects. We will compare students who take sections of the CS2 course using BusTracker projects to students in sections who do not. This will give finer-grained information than simple retention rates on the key issue of whether "relevant" projects have any meaningful impact on attitudes.

Additionally, we also would like to assess how beneficial it is exposing students to advanced CS concepts and technologies, including distributed computing, cyber-physical systems, and real time data processing, early in the curriculum. The courses in which the new programming projects using Web APIs will be introduced will not be concerned with these advanced technologies per se, but the students will be exposed to them indirectly by having to interact with these technologies when working on programming projects. We plan to assess the usefulness of such indirect early exposure by surveying our students once they take more advanced classes. The purpose of this survey will be to determine whether the early exposure through the new projects better prepared our students for the technical concepts and technologies they encounter in more advanced classes and also during summer internships.

Finally, we would consider our project successful if the completed work creates a solid technical foundation for us to submit a TUES II proposal on this topic. For that, three specific goals will have to be reached. These goals naturally build on each other. Firstly, our software infrastructure will have to be solidly engineered to create a viable educational platform. Secondly, we will have to render at least two to three real-time Web data sources accessible through client APIs, so that they could be used in programming projects. We will assign these programming projects in our classes and evaluate their effect on learning outcomes. Finally, we will ensure that our infrastructure (and perhaps some of the APIs) are adopted by peer educators outside of Virginia Tech. Our peers will be welcome to reuse our APIs to access Web data of global relevance (e.g., Yahoo! Stocks) or use our software infrastructure to render their local real-time Web data available for educational purposes. For example, many universities have real-time news portals that can be made available for student projects by means of our software framework. We will consider the proposed work successful if we can find at least three external adopters of our technology.

## 11 Dissemination Plan

We will reserve the domain `www.RealWebAPI.edu` and use it as our gateway both for VT students and our peers at other institutions. One of our design goals is to ease portability, so that CS educators in peer institutions would be able to leverage both our infrastructure and programming projects to help revitalize their introductory classes. The site will include content such as the latest distribution of our educational infrastructure, news about the project, and information about opportunities for undergraduate research.

The gateway site also provides a unique opportunity to disseminate applications (either for the Internet or mobile devices) developed by students. Outstanding course projects can potentially be developed into real applications for public use. We will encourage students to follow up good ideas, perhaps as independent studies or REU projects. The Web-API gateway will then showcase such applications, providing them to the public. To be included at the gateway, an application would have to be submitted to a formal review process. The review process will assess both the project's potential usefulness and quality. Students can even compete for the title of the best application of the semester developed using our infrastructure.

We will target a selected group of colleagues in other institutions to help us pilot both our software infrastructure and some of the APIs. The topics of making CS more appealing and relevant come up often during the professional meetings that the PI and co-PI attend on a regular basis. After our software infrastructure

is built, we will pro-actively contact our colleagues to inform them about the new options they can pursue to revitalize the programming projects in their introductory classes. We will offer help and support in using our software framework to any colleague who volunteers to help us pilot the deliverables of this research.

Additional dissemination of results and knowledge from this work will come in the form of publications and workshops/tutorials at national or international conferences. Appropriate venues to publish the results of this research include SIGCSE, ITiCSE, FIE, CCSC and/or ASEE, all recognized conferences that cover advances in computer science education. Relevant journals include *ACM Transactions on Computing Education* (previously, *JERIC*), Elsevier's *Computers & Education: An International Journal*, *IEEE Transactions on Education*, and Taylor & Francis' *Computer Science Education*. We will prepare tutorials and technical reports (regarding analysis, APIs, and infrastructure development) and make papers and source code available on the Internet.

To increase visibility of our educational infrastructure and encourage broader use of the framework, we will advertise throgh informal channels such as the SIGCSE listserv and contact colleagues who can use the framework in their classes. More formally, we will give a tutorial on using our infrastructure at a major educational conference such as SIGCSE. More generally, both the topics of real-time/real-world data and distributed systems in introductory courses, and the use of "relevant" projects in introductory courses, are important topics for SIGCSE conferences to consider, and are likely to be viable options for special sessions.

## 12 Broadening Participation: Minorities and Undergraduates

In their education-related projects, both Dr. Tilevich and Dr. Shaffer have a history of regularly providing opportunities for undergraduates. The VT Bus Tracker project was essentially started by undergraduates in Dr. Tilevich's class. The work has continued as independent research projects. Dr. Shaffer has had many students work on algorithm visualizations over the past 10 years, including both undergraduate independent study students during the academic year and summer intern students.

The VT CS department generally has had success recruiting African-American female students from neighboring Historically Black College and Universities (HBCU). We have three recent female African American PhD's in computing, (Tracy Lewis, at Radford University, Cheryl Seals at Auburn University, and Jamika Burge at Penn State). We currently are recruiting African-American students from Auburn University, as part of our collaboration with Dr. Juan Gilbert. The success of women in Computer Science in the department is reflected by a high percentage of Ph.D. graduates. Specifically, 10 out of 50 Ph.D. graduates from Spring 2008 to Spring 2011 have been female, which is higher than the national average. We currently have four Latino students in our graduate program (2 PhD students and 2 Masters). One of them, Ricardo Quintana-Castillo, is a recipient of an NSF Graduate fellowship. Dr. Shaffer's NSF-funded AlgoViz projects have supported four graduate students: A.J. Alon (minority), Monika Akbar (female), Michael Stewart (first-generation college graduate), and Eric Fouh (minority).

The CS department participates in the NSF's Research Experience for Undergraduates (REU) and Broadening Participation in Computing (BPC) program, which attracts talented and enthusiastic undergraduate students from our partner minority serving institutions during summer sessions. As part of this program, Dr. Tilevich mentored a female student, SooBeen Kim, and the results of her research project led to a full research paper presented at ICPC 2009 [13]. We will continue to mentor CS@VT REU/BPC students as well as VT MAOP[1] [24] students. Both programs are nationally recognized for involving women and minority students in quality research programs.

PI Tilevich has been mentoring a female CS undergraduate researcher, Kristin Whetstone, who has won two prizes for her research in the 2010 VT Department of Computer Science annual undergraduate research competition. Her research abstract based on the work she has been pursuing under Tilevich's supervision

[1] Virginia Tech's Multicultural Academic Opportunities Program

was accepted to the poster session at the Grace Hopper Celebration of Women in Computing 2010, and also selected for the ACM Student Research Competition (SRC).

## 13 Key Staff

**PI:** Eli Tilevich, Assistant Professor of Computer Science at Virginia Tech, directs the Software Innovations Lab whose mission is to tame the complexity of developing and maintaining emerging complex computer systems through novel software technologies. Prof. Tilevich received his Ph.D. in Computer Science from Georgia Tech in 2005, M.S. in Information Systems from New York University in 1999, and B.A. *summa cum laude* in Computer Science and Mathematics from Pace University in 1997. Tilevich has a proven track record of creating novel software engineering solutions for advanced distributed systems, with more than 40 peer-reviewed technical publications. His recent research has been published in premier venues, including ECOOP'09, ICDCS'09, OOPSLA'09, Middleware'09, and AOSD'10 and '11. In 2011, he is serving as a Technical Program Committee member of top conferences, such as ICSM, GPCE, AOSD, and OOPSLA. Dr. Tilevich led the student team that created the original version of VT Bus Tracker and now continues supervising all the project's development. He will be responsible for constructing and sustaining the proposed educational infrastructure. He will also be in charge of creating new programming projects using the infrastructure for CS2114 Software Design & Data Structures.

**Co-PI:** Clifford A. Shaffer, Professor of Computer Science at Virginia Tech, has been studying the use of simulation and visualization in education since the early 1990s, when he was the PI and lead designer for the highly successful Project GeoSim. This project developed a series of simulations for geography education. He has extensive experience with designing problem solving environments for a number of scientific and engineering applications. Other educational projects involved developing interactive web-based tutorials for undergraduate statistics. He leads development of the AlgoViz Portal, whose goal is to promote use of algorithm visualization in computer science education. He is course coordinator for, and regularly teaches, CS3114 Data Structures and Algorithm Analysis. Dr. Shaffer will teach sections of CS3114 during the life of this project, introducing the BusTracker.EDU software into projects for the course as described in this proposal. He will also help to define the API, with the goal of ensuring that it is easily understandable by undergraduates. He will collect and analyze data from the class to evaluate the contribution of the project.

**Graduate, undergraduate, and intern students:** The project will hire one graduate research assistant to perform mission critical development. The Computer Science department at Virginia Tech has historically had great success in recruiting undergraduate students to contribute on educational research projects, formalized through the VTURCS program. BusTracker has been implemented in part by undergraduate volunteers. We anticipate continuing our tradition by involving a number of undergraduates through independent study projects. We will attempt to increase participation through our successful summer internship program that has been running for a number of years.

## 14 Results from Prior NSF Support

**NSF CCLI Phase 1 Award (DUE-0836940):** *Building a Community and Establishing Best Practices in Algorithm Visualization through the AlgoViz Wiki.* PIs: Clifford A. Shaffer and Stephen H. Edwards. $149,206. for 01/2009 – 12/2010.

**NSF NSDL Small Project (DUE-0937863):** *The AlgoViz Portal: Lowering Barriers for Entry into an Online Educational Community.* PIs: Clifford A. Shaffer and Stephen H. Edwards, $149,999 for 01/2010 – 12/2011.

These projects seek to developed online infrastructure (a website and related community development efforts) to promote use of algorithm visualization (AV) in computer science courses. This is achieved by building a community of AV users and developers. So far, one journal paper [20] and four conference papers [18, 19, 21, 22] have been published relating to this work.

## References


[1] T. Balch, J. Summet, D. Blank, D. Kumar, M. Guzdial, K. O'Hara, D. Walker, M. Sweat, G. Gupta, S. Tansley, J. Jackson, M. Gupta, M. N. Muhammad, S. Prashad, N. Eilbert, and A. Gavin. Designing personal robots for education: Hardware, software, and curriculum. *IEEE Pervasive Computing*, 7(2):5–9, 2008.

[2] T. Beaubouef and J. Mason. Why the high attrition rate for computer science students: some thoughts and observations. *ACM SIGCSE Bulletin*, 37(2):106, 2005.

[3] D. Blank. Robots make computer science personal. *Commun. ACM*, 49(12):25–27, 2006.

[4] L. Carter. Why students with an apparent aptitude for computer science don't choose to major in computer science. *ACM SIGCSE Bulletin*, 38(1):31, 2006.

[5] E. Ciurana. *Developing with Google App Engine*. Springer, 2009.

[6] J. P. Cohoon. An introductory course format for promoting diversity and retention. In *SIGCSE '07: Proceedings of the 38th SIGCSE technical symposium on Computer science education*, pages 395–399, New York, NY, USA, 2007. ACM.

[7] Z. Dodds, C. Alvarado, G. Kuenning, and R. Libeskind-Hadas. Breadth-first CS 1 for scientists. *SIGCSE Bull.*, 39(3):23–27, 2007.

[8] S. Edwards. Using test-driven development in the classroom: Providing students with automatic, concrete feedback on performance. In *Proceedings of the International Conference on Education and Information Systems: Technologies and Applications (EISTA)*, 2003.

[9] A. Fisher, J. Margolis, and F. Miller. Undergraduate women in computer science: experience, motivation and culture. In *Proceedings of the twenty-eighth SIGCSE technical symposium on Computer science education*, page 110. ACM, 1997.

[10] Goolge Inc. General transit feed specification. `http://code.google.com/transit/spec/transit_feed_specification.html`, 2010.

[11] D. Hendrix, D. Umphress, and J. Cross. Designing a first-year project course to engage freshman software engineers: An experience report. In *Software Engineering Education and Training, 2006. Proceedings. 19th Conference on*, pages 25–34, 2006.

[12] J. Hughes and D. R. Peiris. Assisting CS1 students to learn: learning approaches and object-oriented programming. *SIGCSE Bull.*, 38(3):275–279, 2006.

[13] K. Hussein, E. Tilevich, I. I. Bukvic, and S. Kim. Sonification design guidelines to enhance program comprehension. In *ICPC '09: Proceedings of the* $17^{th}$ *IEEE International Conference on Program Comprehension*, 2009.

[14] C. Lewis, M. H. Jackson, and W. M. Waite. Student and faculty attitudes and beliefs about computer science. *Commun. ACM*, 53(5):78–85, 2010.

[15] B. Moskal, D. Lurie, and S. Cooper. Evaluating the effectiveness of a new instructional approach. In *SIGCSE '04: Proceedings of the 35th SIGCSE technical symposium on Computer science education*, pages 75–79, New York, NY, USA, 2004. ACM.

[16] T. P. Murtagh. Weaving CS into CS1: a doubly depth-first approach. *SIGCSE Bull.*, 39(1):336–340, 2007.

[17] L. Rich, H. Perry, and M. Guzdial. A CS1 course designed to address interests of women. In *SIGCSE '04: Proceedings of the 35th SIGCSE technical symposium on Computer science education*, pages 190–194, New York, NY, USA, 2004. ACM.

[18] C. A. Shaffer, M. Akbar, A. J. D. Alon, M. Stewart, and S. H. Edwards. Getting algorithm visualizations into the classroom. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education - SIGCSE '11*, page 129, Dallas, TX, USA, 2011. ACM Press, ACM Press.

[19] C. A. Shaffer, M. Cooper, and S. H. Edwards. Algorithm visualization: a report on the state of the field. *ACM SIGCSE Bulletin*, 39(1):150–154, 03/2007 2007.

[20] C. A. Shaffer, M. L. Cooper, A. J. D. Alon, M. Akbar, M. Stewart, S. Ponce, and S. H. Edwards. Algorithm visualization: The state of the field. *ACM Transactions on Computing Education*, 10:1 – 22, 08/2010 2010.

[21] C. A. Shaffer, T. L. Naps, and E. Fouh. Truly interactive textbooks for computer science education. In *Submitted to 6th Program Visualization Workshop*, Darmstadt, Germany, June 2011.

[22] C. A. Shaffer, T. L. Naps, S. H. Rodger, and S. H. Edwards. Building an online educational community for algorithm visualization. In *Proceedings of the 41st ACM technical symposium on Computer science education - SIGCSE '10*, page 475, Milwaukee, Wisconsin, USA, 03/2010 2010. ACM Press, ACM Press.

[23] E. H. Turner, E. Albert, R. M. Turner, and L. Latour. Retaining majors through the introductory sequence. *SIGCSE Bull.*, 39(1):24–28, 2007.

[24] Virginia Tech. The multicultural academic opportunities program (MAOP), December 2009. `http://www.maop.vt.edu/`.