Mentoring Undergraduate Students Preparing for Graduate Study in Engineering-A CREATE Case Study

Lane Thames and Randal Abler
Georgia Institute of Technology, lane.thames@gatech.edu, randal.abler@gatech.edu

Abstract – **This paper discusses the Collaborative Research Experiences in Advanced Technology and Engineering (CREATE) undergraduate research program and one of its associated research projects. CREATE is a program that targets undergraduate students from non-research based universities who are interested in pursuing graduate study in engineering. CREATE is a short-term project that takes place during the summer semester. Since it is a short-term project and the students have little experience with engineering research, the authors posit that a research environment that conforms to the student's incoming skill-set is needed so that the time needed for the student to learn the experimental environment can be reduced. A solution to this need is the introduction of a rapid research and development environment using software plugin architectures. This paper describes the plugin architecture and two types of plugin architectures related to network health monitoring and network security that can be used as the basis of a rapid research and development environment.**

Index Terms – Collaborative Undergraduate Research, Rapid Research and Development, Network Security, Network Monitoring

INTRODUCTION

During the summer of 2007, the Georgia Institute of Technology's Savannah campus initiated its first phase of the Collaborative Research Experiences in Advanced Technology and Engineering (CREATE) undergraduate research program. The CREATE program invites undergraduate sophomores, juniors, and seniors, especially encouraging students from non-research based institutions, to participate in advanced engineering research. Students who participate in the project gain first hand experience in performing engineering research by working directly with faculty and graduate student mentors on a research project, by attending faculty research seminars, and by giving a poster presentation describing their research as a final deliverable. Further, the students receive professional training by participating in tours of local engineering facilities, by attending resume development seminars, and by attending seminars that discuss the process of applying to and paying for graduate school. In this paper, we provide a case study for one of the associated CREATE research projects. Specifically, we discuss a research project in network security. We describe the design of the phase 1 research project illustrating how we envisioned the teaching and mentoring process for a student with a non-research background. The paper describes the obstacles that were faced with respect to the original design specifications of the project related to the student's ability to conduct the engineering research and how the design has changed in preparation for the phase 2 project that will occur in the summer of 2008. The major obstacles faced by the student were a result of 1) a short time frame for doing the research (10 weeks) and 2) a steep learning curve with respect to the research topic and the necessary prerequisite technical knowledge needed to work within the experimental setting. The design of the phase 2 project incorporates a technique aimed to alleviate some of these obstacles. The technique is the development of a Rapid Research and Development (RRD) methodology using the concept of software *plugins*. This technique will be applicable for engineering and computer science students who need an environment that allows for rapid software development and prototyping of ideas needing to be studied in a research setting. Specifically, the plugin architectures discussed in this paper can be used for advanced research in monitoring a network's health or advanced research in network security in the areas of firewall technology, attack detection technology, and information distribution technology.

CREATE PHASE I APPROACH

During the first phase, the following approach was taken.

1. The student was introduced to the topic with a series of short lectures.
2. The student was taught the process of performing a literature review for the purpose of conducting engineering based research. The student was shown how to access academically sponsored databases and how to find relevant and worthy publications.
3. The student was tasked with performing the literature review and producing a report of the review. During this iterative process, the student was guided by her mentors such that she focused on the particular topic of interest.

4. After observing previous work and relating it to the topic of interest, the student was tasked with setting up her experimental environment, conducting the experimentation, and producing an analysis of the research results.
5. The student finished the project by communicating her results during a poster session that was open to the campus community.

One will recognize this approach as the classical approach to academic research. This includes a literature review, laboratory experimentation, evaluation and analysis of the experimentation, and communication of the results. The classical research approach works well for projects that have respectable time constraints. However, the CREATE project is a *short-term* research project for students with little or no research experience, and we observed that step 4 requires careful consideration during a short-term research project such as this. Why? We observed that the student spent a significant portion of her time during this step learning how to work ***with*** the experimental environment. With a short-term project, this "significant portion" of time can account for a large percentage of the available research time, and this is undesirable. In this case, about 60-70% of the student's time that was allocated for the research project was spent learning the experimental environment, which included learning Matlab and Linux operating system concepts. The 60-70% value was a result of the short time frame that was available for the entire project. Of course, with longer term projects, this percentage decreases. The percentage is calculated as the time spent learning the environment divided by the total time. So, if the total time increases, the percentage decreases.

Learning to work with the experimental environment is not research. It is, in general, not meaningless time spent as it provides a positive learning experience for the student. However, when the focus is on research, the ultimate goal should be learning ***from the research results***, not learning ***of the research's technical environment***. Therefore, for short-term research projects involving students with limited technical backgrounds and/or research experience, a solution is needed that provides a flexible, easy-to-use environment for experimentation of research topics. For completeness, the next section discusses the research objectives and results for a particular CREATE research project. Then, we discuss a possible solution for the dilemma of short-term research projects with limited time constraints.

PHASE I RESEARCH RESULTS

One of the CREATE research projects was a study of network security. The student's task was to explore various artificial intelligence algorithms that could classify computer intrusions, i.e. acting as an intrusion detection system. The algorithms studied were Naïve Bayesian Learning Networks (NBLN) [1] and Artificial Neural Networks (ANN) [2]. To evaluate the classification accuracy and computational performance of these algorithms, the Knowledge Discovery and Data mining (KDD) intrusion detection database [3] was used. The KDD intrusion detection database is a highly referenced dataset for evaluating intrusion detection systems. This dataset contains 42 variables for each row in the dataset. One of the variables labels the data in the row as being related to a normal network connection or related to a network connection that represents an attack. The objective of the research was to evaluate the NBLN and ANN using a reduced feature subset of the KDD dataset. The reduced feature subset included 8 of the 42 KDD dataset variables. MATLAB was used for evaluating the classification algorithms. Two primary performance metrics were defined to evaluate the algorithms: classification accuracy and computational performance. The dataset included a "label" variable that described the row's data as either being an intrusion or being normal. To calculate the classification accuracy, the algorithm's output was compared to the row's label. If the output equaled the label, then it was a correctly classified result. Otherwise, it was incorrectly classified. The computational performance metric was defined as the amount of time needed to classify a single record. The results are the following. The NBLN had a classification accuracy of 93.454% and a computational performance of 0.291 seconds/record. The ANN had a classification accuracy of 72.426% and a computational performance of 0.008 seconds/record. As these results show, the NBLN had much better classification accuracy, but the ANN had much better computational performance. Once the research project was complete, the student communicated her research results during a poster session that was open to the Georgia Tech community.

CREATE PHASE II APPROACH

The observation made during the first phase of the CREATE project was that the student spent a significant proportion of the research time learning the technical details of the experimental environment. In this case, the student had to learn the technical details of programming Matlab functions. The student came into the project with experience in the C++ programming language. However, the student had no experience with computer networking, artificial intelligence, or network security. Learning the high-level concepts of networking, network security, and artificial intelligence took place during the literature review. However, once the experimental aspect of the research began, the student had to learn the details of Matlab. This took a significant amount of time for the student. Another course of action could have been allowing the student to develop the software using C++. However, the student had little experience with artificial intelligence and would have needed to develop the code necessary for the ANN and NBLN. On the other hand, Matlab provided toolbox functions for the needed artificial intelligence algorithms. So, there was a trade-off—should the student spend time writing code about abstract concepts (i.e. the artificial intelligence) in a language she understood, or should she spend time learning a new language that provided the needed functionality? This type of observation can be made time after time when teaching research to undergraduate students.

This leads to our problem statement: "How does one construct a research environment such that an incoming student with little or no research background can produce meaningful research results when the research project has strict, short-term time constraints?" A solution that we propose is based on ***software plugin technology***. This solution is only applicable to research that requires software development as the underlying research environment. However, the *concept* of plugin technology could possibly be applied to other research areas. The goal is to produce a research setting where the environment conforms to the student's skill set instead of the student needing to conform to the research environment thereby reducing the percentage of time spent learning the experimental environment and increasing the percentage of time performing actual research. For example, if the student's skill set includes a particular programming language, the environment should offer the ability for the student to use that particular language instead of needing to learn a new language for performing the research. The solution needs to be highly modular and extensible, and the solution should offer multiple research topics for which the student can choose. We posit that software plugin technology provides an excellent platform that can be used as the solution for this problem. Software plugin architectures are highly modular and extensible, they are easy to use because they do not confine the user to a particular prerequisite skill set (which reduces the student's experimental environment learning time), and their "pluggable" nature provides an avenue for offering many research topics to the student.

I. Software Plugin Technology

Software plugin technology is not new. It has been applied to many types of software architectures in the commercial domain such as email and web based technologies. However, we have not found any references to others using it as the basis of an undergraduate research environment. Figure 1 illustrates a generic view of a software plugin architecture. A plugin can be defined as a software application that interacts with some *host* application using a well defined Application Programming Interface (API) that facilitates inter-process communication. The host can provide services to the plugin, the plugin can provide services to the host, or a mixture of the two types can be used. Extensibility can be achieved because of the well defined API. Plugins can be added to the application, removed from the application, or upgraded within the application with little effort. Further, because the plugin only needs to adhere to the API, the technology needed for the plugin's implementation can, in many cases, be very flexible, i.e. it can be written in a variety of programming languages. As shown in Figure 1, a software plugin architecture can have N unique plugins operating at any given time, and they communicate with the host application via the API.

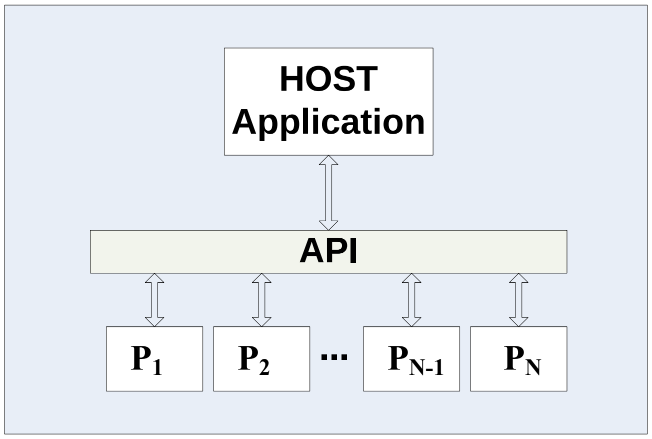

FIGURE 1
SOFTWARE PLUGIN ARCHITECTURE

Our research environment during the second phase of the CREATE project, which will occur during the summer of 2008, will consist of research in the area of network health monitoring or network security. The topic chosen will depend on the incoming student's interests related to topics in these two areas. To evaluate the idea of RRD with plugin technology, we will be using the Nagios [4] network monitoring open-source software or the Distributed Firewall and Active Response (DFAR) [5],[6] open-source software. Both of these software packages use the plugin based architecture and provide extensible environments with no constraints on the underlying plugin implementation techniques other than adhering to the defined APIs.

II. Network Monitoring Research with a Plugin Architecture

Nagios is an open-source software package that is used to monitor the health of networks. This includes the monitoring of any network device, server, host, or network service that the user deems necessary to ensure correct operation of the networked system. Figure 2 illustrates the Nagios architecture. The architecture consists of the core system, an event notification system, a configuration system, the plugin API, the plugins, and a web interface. The web interface can be used to set various configuration parameters and to view event notifications. With the Nagios architecture, the core uses the services provided by the plugins. The Nagios system is meaningless without the plugins as they provide the network monitoring functionality. Event notification is triggered when a plugin communicates to the core that a particular system being monitored has malfunctioned, i.e. a network device is offline or a particular network service is not functioning correctly. Event notifications are normally sent to the system administrator via email or text messaging. The Nagios plugin architecture provides a simple API for plugins to communicate with the core. The API simply states that the plugin should generate a particular numeric "exit" code when the plugin's internal code has finished its operations. These exit codes communicate the plugin's monitoring result to the core.

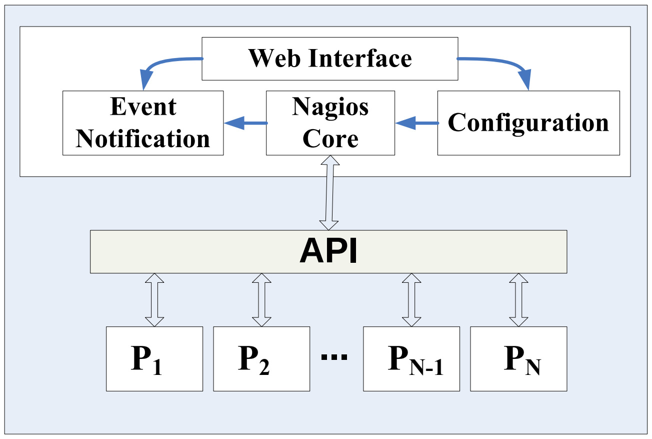

FIGURE 2
THE NAGIOS NETWORK MONITORING ARCHITECTURE

The API defines the exit codes as the following: an exit code of 0 implies that the monitored entity is in an OK state, an exit code of 1 implies that the monitored entity is in a WARNING state, an exit code of 2 implies that the monitored entity is in a CRITICAL state, and an exit code of 4 implies that the monitored entity is in an UNKNOWN state. A primary advantage of this architecture is that any programming language that provides an "exit" construct (and most do) can be used to develop the necessary plugins. From the RRD perspective, the student can focus on a network monitoring research topic of interest and can quickly perform research, experimentation, and analysis because the student can use one of their previously acquired skill sets, i.e. a programming language in which they are already familiar.

III. Network Security Research with a Plugin Architecture

The Distributed Firewall and Active Response (DFAR) architecture is a network security system that provides abilities for distributed firewalls to dynamically update their firewall rules (or access control lists) in response to computer attacks. This ability is provided by autonomous detection agents that monitor a local host in real time and observe computer attacks. Once attacks are observed, the detection agents generate information that is processed by the local host such that new access control rules can be added to the local host's firewall. Further, the local host communicates these results to its neighbors so that they can also add new firewall rules related to the observed attacks. Figure 3 illustrates the DFAR architecture. DFAR is composed of 4 primary management modules: *Firewall Management Module*, *Policy Management Module*, *Distribution Management Module*, and *Autonomous Detection System (ADS) Management Module*. The architecture contains two specification interfaces: *Firewall Rule Specification Interface (FRSI)* and *Policy Description Specification Interface (PDSI)*. A complete discussion of the DFAR architectural specifications can be found in [5],[6]. As seen from Figure 3, the DFAR architecture is highly modular and follows the plugin architecture. Specifically, the Autonomous Detection Agent (ADA) modules are plugin-based components and can be developed with any programming language desired by the implementer. The only requirement is that the ADA adheres to the rules defined by the PDSI. The PDSI defines the API by which the ADA modules communicate with the system's other modules. The goal of the ADA is to detect computer attacks generated by some anomalous host. Once the attack is detected, the ADA communicates the detection information to the policy management module through the PDSI. The API for this system is simply that the ADA supplies the PDSI with the Internet Protocol (IP) address of the anomalous (attacking) host. The FRSI is the system's other interface, and it defines the API between the policy management module and the firewall management module. This API is simply the IP address of the attacking host, a policy action value, and a rule operation. The architecture defines the policy action value to be "DENY" and the rule operation to be insert, modify, or delete, which corresponds to inserting a new firewall rule, modifying an existing rule, or deleting an existing rule. These interfaces allow the architecture to be highly extensible whereby many types of ADAs can be active on a system, and the local host does not need to use a particular type of firewall.

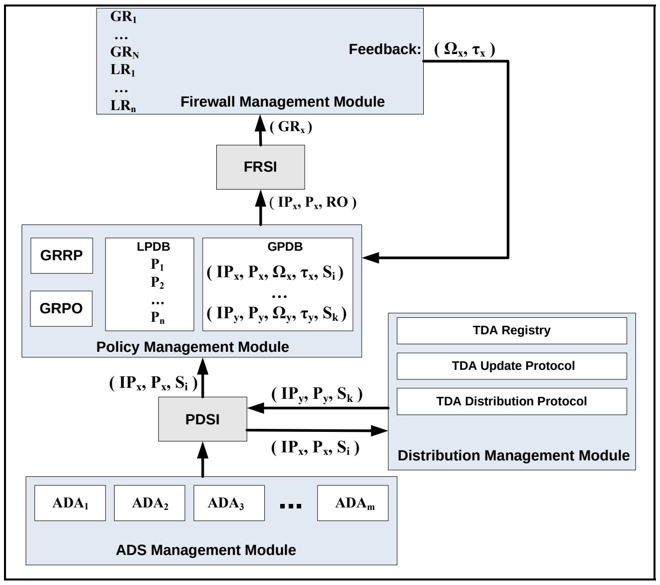

FIGURE 3
THE DISTRIBUTED FIREWALL AND ACTIVE RESPONSE ARCHITECTURE

With respect to RRD, the DFAR architecture provides several opportunities for students to produce significant research results with minimal time needed to learn the experimental environment. Research opportunities provided by the DFAR architecture include the following: computer attack detection theory, intrusion detection theory, database theory, firewall technology, firewall optimization, information distribution theory, and cryptographic theories such as information confidentiality, authenticity, and integrity.

In summary, our approach during the second phase of the CREATE project will be similar to the first phase. However, step 4 will be designed such that the student will use one of the plugin based architectures described in this paper. We believe that the plugin architecture will provide an experimental environment that conforms to the student's skill sets such that minimal time will be required to learn the experimental environment. Further, the plugin architecture will provide an array of research topics for the student to choose. We believe that this approach will significantly enhance the student's overall research experience.

CONCLUSION

This paper discussed the Collaborative Research Experiences in Advanced Technology and Engineering (CREATE) undergraduate research program, which is an undergraduate research program that targets undergraduates who have little or no experience with undergraduate research in engineering but who are interested in pursuing graduate study in engineering. CREATE is a short-term research program, and the authors observed that students with limited research and technical engineering backgrounds need a research environment that is conformable to their skill-sets when the project has short-term time constraints. The paper introduced the concept of rapid research and development and the use of software plugin architectures to provide a conformable environment such that rapid research and development can be achieved. During the second phase of the CREATE program, one of the associated research projects will use a plugin architecture as the basis of the student's experimental environment so that the concept of rapid research and development can be evaluated.

ACKNOWLEDGMENT

The authors would like to thank Dr. Elliot Moore of the Georgia Institute of Technology. Dr. Moore is the principal investigator of the Collaborative Research Experiences in Advanced Technology and Engineering research program at the Georgia Tech Savannah campus.

REFERENCES

[1] Pearl, J., "Bayesian Inference", *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann Publishing, 1988, pp. 29-62

[2] Mitchell, T., "Artificial Neural Networks", *Machine Learning*, McGraw-Hill Publishing, 1997, pp. 81 - 126

[3] The Third International Knowledge Discovery and Data Mining Tools Competition, KDD Cup 1999 Dataset, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[4] The *Nagios* open-source network monitoring system, http://www.nagios.org/

[5] Thames, J.L., Abler, R.T, "A Distributed Firewall and Active Response Architecture Providing Preemptive Protection", *Proceedings of the ACM Southeast Conference*, March 2008

[6] Thames, J.L., Abler, R.T., "A Distributed Active Response Architecture for Preventing SSH Dictionary Attacks", *Proceedings of the IEEE Southeast Conference*, April 2008

AUTHOR INFORMATION

Lane Thames, PhD Candidate, Georgia Institute of Technology, lane.thames@gatech.edu

Randal Abler, Professor, Georgia Institute of Technology, randal.abler@gatech.edu