New Orleans JAZZED about Engineering Education

Paper ID #16031

Incorporating Service-Oriented Programming into the Computer Science Curriculum

Dr. Xumin Liu, Rochester Institute of Technology

Xumin Liu received the PhD degree in computer science from Virginia Tech. She is currently an Associate Professor in the Department of Computer Science at Rochester Institute of Technology. Her research interests include data management, data mining, web services, semantic web, business process management and mining, social computing, and cloud computing. Her research has been published in various international journals and conferences, such as the International Journal on Very Large Data Bases (VLDB journal), IEEE Transactions on Service Computing (TSC), International Journal of Web Service Research (JWSR), IEEE International Conference on Web Services (ICWS), and International Conference on Service Oriented Computing (ICSOC). She frequently serves as a program committee member for various international conferences and review for various journals, including ACM transactions and IEEE transactions, and others.

Dr. Rajendra K Raj, Rochester Institute of Technology

Rajendra K. Raj is a Professor of computer science at the Rochester Institute of Technology. His current research interests span large scale data management, distributed systems, and privacy/security, especially related issues in cloud data management and mobile computing, and applied to a variety of domains including healthcare, finance and critical infrastructure protection. Dr. Raj also works on computer science education issues including curriculum design and program assessment. Prior to RIT, he served at a financial services firm, where he developed and managed leading edge global distributed database infrastructures for a variety of financial applications. He earned his Ph.D. from the University of Washington, Seattle.

Dr. Chunmei Liu, Howard University

Dr. Chunmei Liu is currently a Professor of Computer Science Department at Howard University. She received her Ph.D. in computer science from The University of Georgia. She joined Howard University in 2006 as an Assistant Professor. Her research interests are algorithms and computational biology.

Dr. Alex Pantaleev, SUNY Oswego

Alex Pantaleev received a B.A. degree in computer science from the American University in Bulgaria, Blagoevgrad, Bulgaria, in 2003, and M.S. and Ph.D. degrees in computer science from the Ohio State University, Columbus, Ohio in 2007 and 2008, respectively. He is currently an Associate Professor in the Department of Computer Science at the State University of New York, Oswego.

Incorporating Service-Oriented Programming into the Computer Science Curriculum using Course Modules

Abstract

In modern computing and engineering programs, new course materials need to be added regularly in a flexible manner. The concept of *course modules* has been suggested as one approach to doing this; a course module, which is a self-contained unit of curriculum such as a lab or teaching component, can be included into existing courses without requiring substantial course or program modifications. In this paper, the authors describe their experiences in incorporating new curricula into computer science and engineering curricula at their three institutions, including Rochester Institute of Technology, Howard University, and SUNY at Oswego. The relatively new paradigm of Service-Oriented Programming (SOP) was introduced into their programs using course modules. SOP is a relatively new programming paradigm in which software applications are wrapped as web services, which can be used as building blocks for developing new software applications. Leveraging ubiquitous Internet access and intensive standardization efforts, SOP boosts software and resource reuse. Driven by the benefits of SOP, the number and variety of web services are currently offered online. ProgrammableWeb, one of the largest public API registry, lists over 14,000 web services in various categories, such as weather, social, travel, education, music, and animation. It also lists over 6,000 software applications developed by using public web services, i.e., service mashups. Although such web services represent an attractive opportunity for incorporation in undergraduate CS and SE curricula, few if any of today's freshman and sophomore students in these majors know anything about the concepts underlying web services. When available, courses covering aspects of web services are primarily offered to undergraduate seniors or graduate students. To include SOP in undergraduate CS and SE curricula, this paper describes the design and development of these SOP course modules, their implementation in their three institutions over the past three years, assessment and evaluation of the implementation, and some lessons about the incorporation of new coursework into computing and engineering curricula.

Introduction

In highly dynamic developing fields such as computer science (CS), it is crucial that CS curricula incorporate new course materials in a flexible manner. To ensure new course materials are developed for flexible incorporation, the notion of course modules comes in handy. A course module, which is a self-contained unit of curriculum such as a lab or teaching component, can be included in existing courses without requiring substantial course or program modifications. This paper describes the use of course modules to introduce the concepts of Service Oriented Programming (SOP) at multiple levels of the undergraduate CS curriculum in three participating institutions, including Rochester Institute of Technology, Howard University, and SUNY at Oswego.

SOP is a relatively new programming paradigm in which software applications are wrapped as web services, which can be used as building blocks for developing new software applications[1,7]. Leveraging ubiquitous Internet access and intensive standardization pushed by the W3C, SOP helps to boost software and resource reuse[5,9]. SOP also helps to deliver a variety of business functionalities, for example, mapping, routing, and other travel services. Large numbers of web services are currently offered online, with over 10,000 web services covering eight different domains are registered with an open web service registry, ProgrammableWeb. Although such web services represent an attractive opportunity for incorporation in undergraduate CS curriculum, few if any of today's freshman and sophomore students in these majors know anything about the concepts underlying web services. When available, courses covering aspects of web services are primarily offered to undergraduate seniors or graduate students. Therefore a compelling case can be made for the inclusion of SOP in undergraduate CS[6].

The authors have addressed the SOP curricular gap through the design, creation and use of two SOP course modules: (a) an introductory module aimed at the freshman courses such as CS 2, and (b) a mid-level module for courses such as Programming Language Concepts or Computer Networks. A third course module that aimed at the senior level elective course has also been designed but is beyond the scope of this paper. As web services are often a brand new concept to many students (and some faculty), the course modules rely heavily on examples to illustrate how to develop and consume web services in different development environments and in different application domains[2,3]. Being self-contained, the two modules can also be included in other suitable courses.

Over the past couple of years, the authors have incorporated these two modules into their CS coursework at their three institutions. This paper describes the design and development of the SOP course modules, their usage in their three institutions, and early lessons from these offerings. Based on these experiences, the paper also lays out an approach for flexible incorporation of new coursework into computing and engineering curricula.

Background – What is SOP?

In the Service Oriented Programming (SOP) paradigm, software systems of different sizes can be wrapped up as web services, which can be reused by other programmers regardless of heterogeneous settings, such as operating systems, development platforms, programming languages, and data structures. SOP lays the foundation for developing loosely coupled, independent, and distributed software components and helps to boost software reuse.

Key to SOP are web services, which are self-contained network-accessible software units that complete tasks, solve problems, perform transactions, and so on[4]. These services are different from web-based applications, which are for human consumption via GUIs. Web services are the APIs consumed via program invocation; examples of web services include Google Map APIs, YouTube APIs, and Expedia hotel APIs.

Compared to traditional software APIs or libraries, web services provide two additional benefits: global accessibility and standardization. First, web services leverage the powerful communication paradigm of the web and the ubiquitous HTTP/SMTP transport protocols. Services may be accessed globally and via firewalls. Web services also provide a lightweight approach to offering and reusing software applications. Unlike traditional software libraries that need to be downloaded and included in a project package, web services are invoked by exchanging well-formed messages on the Web. Nothing but client-stubs needs to be included in the project package. The user-side burden can be further relieved when invoking RESTful services, where REST design pattern is used in web service development and deployment. User-side programs only need to send a http request and parse the response to consume a service. Second, web services have benefited from efforts by multiple organizations to standardize their interactions with users. The three core web service standards are Web Service Description Language (WSDL) for service description, Universal Description Discovery and Integration (UDDI) for service publishing and discovery, and Simple Object Access Protocol (SOAP) for service invocation. Other aspects of web services also have standards such as Business Process Execution Language (BPEL) for web service composition, WS-Security for enhancing secure interactions, Web Service Conversation Language (WSCL) for coordinating service invocations, WS-Transaction for managing web service transactions, and WS-Notification for notifying changes of a web service to its subscribers. For the purpose of addressing interoperability issues, all the web service standards are XML-based. All of these benefits have helped web services gain attention and establish its importance in modern software development. Web services also play an essential role in online business delivery and mobile apps development. As of August 2015, ProgrammableWeb, one of the largest online web service repositories, indexes more than 13,000 public web services.

Given the above benefits, SOP provides a new methodology of solving problems. When dealing with a complex problem, instead of building solutions from scratch, software developers can look at what are available on the web and use them as building blocks. This software reuse helps to avoid repeated work and minimize development costs.

An example SOP application

Suppose we were to assign a student project to build an application that provides local hotel information. It provides an interface for users to specify their query conditions, such as an address, the radius, and the type of hotels. Based on the condition, the system returns users a list of matched hotels near to the given address within the specified radius. For additional utility, the application also provides other related information, such as local weather, nearby airports, gas stations, and driving directions.

If the students needed to use traditional programming methods, i.e., without using web services, it would take great amount of resources and programming effort to develop such an application. The development may consist of several steps. First, the application would need to collect the data related to hotels, weathers, maps, driving routing, and so on. The data collection could be done manually, or could utilize online information by

identifying and crawling suitable websites. Second, the application would need to create a database to store and manage the data. Given the scope of the system, i.e., regional, national, or even international, a large scale or distributed database might be needed, which can introduce significant development and maintenance overhead. Finally, the application would need to design and implement several logical steps including the determination of the distance between two locations, ranking hotels based on locations, querying airport information, and identifying nearby gas stations. Moreover, after this application has been developed, it cannot be easily reused by another software application due to potential interoperability issues, such as the difference of programming language and development platforms.

Under the SOP paradigm, the development becomes easy and flexible. The problem is first decomposed into several components: (1) a *hotel query* component that takes an address, a radius, and the hotel type as input and returns a list of hotels nearby and related information such as names, addresses, and contact numbers; (2) a *weather query* component that takes an address as input and returns the local weather information such as temperature and wind; (3) a *gas station query* component that takes an address as input and return a list of gas stations; (4) an *airport query* component that takes an address as input and return the local airport; and (5) a *route query* component that takes two addresses as input and returns the driving directions between them. To build this application, students can investigate existing web services that can be used to develop these components. Examples of such web services include Expedia APIs, underground weather API, and Google directions APIs. The development process then consists of finding suitable services and invoking them in a proper order. Using existing services significantly decreases the requirements of the resources and programming skills on software developers and shortens the development cycle. In fact, the major component of this application can be designed as a lab assignment, which undergraduate students can finish in a lab session; we discuss this in more detail below in Section II.D.

As a side-benefit of SOP, the developed application can be made more reusable if it is wrapped as a web service.

The SOP Course Modules

A course module is a collection of curricular materials that typically addresses one specific course learning outcome. It removes barriers by providing adoptable materials, summarizes required background expertise, and makes it easier to gain institutional acceptance[8]. Each course module is independent and a faculty member can individually select a course module to experiment with and adopt it to his or her own courses. Course modules thus represented an excellent approach to incorporating curricular materials into multiple institutions in a flexible manner.

Table 1 shows the components of our course modules. To teach SOP and incorporate the developed course modules into CS/SE curricula, we designed two course modules to be

offered at two levels—introductory and mid-level in both CS and SE. These course modules are discussed next.

Table 1 Components of a course module

Component	Brief Description
Overview	Description of module, prerequisite knowledge, and module learning outcomes
Rationale	Motivation for the module
Recommended use	Recommendations for typical usage
Slides	Module content for lectures or independent learning activities
Sample questions	For use in low-stake quizzes
Labs/assignments	For hands-on experiences in solution design, implementation, and verification
FAQ	Answers to students? frequently asked questions
Readings	Introductory or supplementary materials referenced in the module
Links	Pointers to online and other materials used in the module
Module evaluation	Assessment tools to measure learning and module effectiveness

Course module 1 – Introductory level

This course module, referred to as CM1, targets CS2, the second course in the problem solving sequence typically required for first year students in several computing majors such as Computer Science, Software Engineering, Computer Engineering, and Computational Mathematics. Students in the class are already prepared with basic idea of computational thinking, programming languages and techniques, elementary data structures, and some basic problem solving methods. They continue to learn those topics with more complex features and typically from object-oriented perspective. This makes CS2 an ideal choice as the first course to introduce SOP concepts.

CM1 aims to students to SOP as a new programming methodology, stimulate their interests, and motivate them to continue their learning in SOP from other SOP course modules and outside of classroom. Therefore, it has a slight touch on the Web Service technologies and puts the major focus on the motivation of SOP, its advantages over traditional programming paradigms, its current status, and real world examples. Following this line, CM1 is designed as a mixture of problem solving questions, in-class demos, lectures, and lab assignments. It can be taught in one or two lectures and a lab session, followed by the topics of object-oriented programming. The lecture can start with a problem solving question, where students are asked to provide a solution to reuse a piece of programming code they write previously in CS2 and several complications/restrictions are gradually added to the problem gradually, such as mismatched data structure, mismatched variable names, and different development

languages. This gives students an example that can help them understand the limitation of traditional programming methods. SOP can be then introduced as the solution to address the limitation. A demo is designed where the code of sorting algorithm is wrapped as a web service and an application outputs the sorting result by remotely invoking the service through the network.

After this demo, instructors can present basic concepts, development history, and features of SOP, mixed with more demos on the usage of current public web services in developing a complex application. The basic concepts include web services, standards, such as Web Service Description Language (WSDL) for describing a web service's functionality and generating client-side stubs, Simple Object Access Protocol (SOAP) for formatting messages exchanged between services and client programs, which carries invocation arguments and results, and Universal Description, Discovery and Integration (UDDI) for providing a third-party support for service providers to advertise services and for service users to query desirable services. Representational State Transfer (REST) will be also covered, which is a design pattern where web services are represented as resources on the web. REST has been adopted by many service providers and has gained significant popularity now. After reviewing web service development history and current stage of SOP, demos on public and popular web services can be used to stimulate student interests and engage them in class. Examples of these services are Google Map APIs, Underground Weather APIs, and Expedia Hotel APIs. The demo cannot only showcase how to invoke these services from programs, but also demonstrate how web service technologies turn developing a complicated application much more feasible and easier through software reuse than developing from scratch.

The lab designed for CM1 aims to help students gain hands-on exercise of SOP programming and reinforce what they have learned from the class. It consists of two sections: problem solving and implementation. During the problem solving section, students are given the description of an application and the documentation of several related web services. More specifically, students are asked to develop a hotel query application by using public web services. The application returns a hotel for a given city, the driving directions from the current address to the hotel, and the weather of the hotel area. Students are asked to read the documentation and discuss in groups the functionality of the web services, how to invoke them, and how to use them to develop the application. After getting feedback from the instructor, students implement their solution through coding. Step-by-step instructions on invoking web services are given to minimize the assistance needed from instructors.

Course Module 2---Mid-level

This course module, referred to as CM2, targets a mid-level programming course. For this paper, let us assume this course to Programming Language Concepts (PLC) as it is a common course in many CS programs; CM2 is flexible enough that it can be used in other mid-level courses. The PLC course typically examines various high-level programming languages and programming paradigms, focusing on the underpinning concepts rather than the language details. As SOP is relatively a new programming

paradigm and students in the class may have already had basic knowledge of SOP covered by CM1 in CS2, CM2 covers SOP with more depth and puts its focus on the principles and supporting infrastructures.

CM2 compares SOP with existing programming paradigm such as imperative, functional, logic, and object-oriented; it emphasizes SOP's software reuse benefit. The topics in CM2 center around Service Oriented Architecture (SOA), the underlying architecture that addresses interoperability issues of the interactions between services and users; the syntax of XML, the modeling language that facilitates interactions between heterogeneous software systems; and an in-depth exploration of web service standards and design pattern, including WSDL, SOAP, and REST.

The lecture slides of CM2 can help instructors teach SOA through the explanation of three interaction roles, including service providers, service users, and service registries; the corresponding operations, including *publish*, *find*, and *bind*, shown in Figure 1.

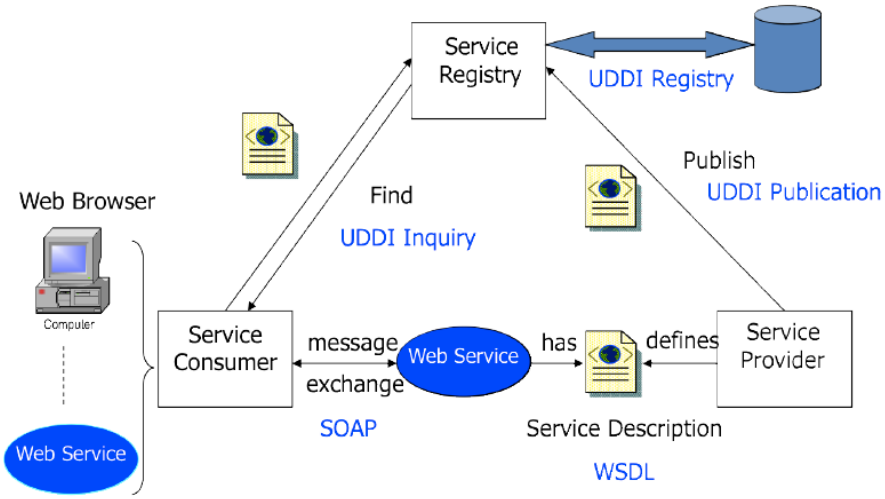

Figure 1: Service-Oriented Architecture

Web service providers have their own web services and implement business logic of the services. They host the services on the web through web servers such as Apache Tomcat and GlassFish server and control the access to these services. Examples of web service providers include Microsoft providing Bing Search APIs, Google providing Google Map APIs, and Expedia providing hotel Search APIs. Web service users are the client-side applications that need some certain functionality. They look for suitable services that can fulfill the need. Once a web service is identified, service users set up the interaction links with web service providers, i.e., performing the bind operation, and call the services through sending invocation information and receiving responses. Web service registries provide a search engine where service descriptions can be published and searched. They serve as the mediator between service providers and service users and facilitate cross-organization and cross-platform integration of software systems. The *publish* operation performed by service providers and *find* operation performed by service users can be automatic through a UDDI registry or semi-automatic through a web service search

engine, such as ProgrammableWeb. Including a service registry in the architecture is a unique feature of SOP compared to other programming methodologies. It allows service users, i.e., programmers, to fully leverage a dynamic software repository on the web and automate the searching process.

CM2 contains the lecture slides of XML, WSDL, SOAP, and REST to teach students or review the topics with them if they have learned from CM1. Compared to CM1, it covers more topics and in more depth, such as the structure of WSDL document and the content of service description; the structure of SOAP message; the usage of SOAP header to enforce security and coordinate the invocation orders among multiple operations. It is also interesting for students to learn how SOAP enables asynchronous invocations. That is, once a client side sends out the invocation message to call a service, it does not always have to wait for the response to proceed to the next step. This is different from using software libraries or traditional Remote Process Call (RPC). Such a feature of SOAP realizes a loosely coupled programming binding and extends the flexibility of software application integration.

The lab designed for CM2 focuses on developing, deploying web services, and understanding their behavior. Students are given step-by-step instructions on how to create a web service project, design and implement the logic, configure web service, host the web service, and test it from client side. After that, students are asked to design their own services that contain multiple operations. Once the service is deployed and tested, students are asked to develop a client side application which needs to combine the operations together to implement a complicated process. Depends on the time allocated to the lab session, the scale of the web service can vary from a simple one, such as a calculator providing addition, subtraction, multiplication, and division operations, to a complicated one, such as a product management system, which allows to query and purchase products.

For this lab, we also designed and developed a visualization tool that students can use to compose clients to already existing services by means of simple drag-and-drop operations. For instance, a student may develop a calculator web service, then use the visualizer to connect to her services and feed the output of one service to the input of another without writing code. Students can use the visualizer as both a mechanism to quickly test their services, and as a tool to facilitate better understanding of the concepts.

Implementation

Over the past three years, we have deployed the two SOP course modules at three institutions. As shown in Table 2 and Table 3, the two course modules were taught in the typical CS programming classes, such as CS2, CS3, Programming Language Concepts, Abstract Data Types and Programming Methodology, and Web Services. For classes where we were not the instructors, we got the support from the instructor who allowed us to teach as a guest instructor, or taught the course module him/herself.

Table 2: CM1 deployment at three institutions

Institution	Term	Course
Rochester Institute of Technology	Fall 2003: 2 sections	CS2
Howard University	Spring 2012: 1 section Spring 2013: 1 section Spring 2014: 1 section	CSIII
SUNY at Oswego	Spring 2015: 2 sections	Abstract Data Types and Programming Methodology

Table 3: CM2 deployment at three institutions

Institution	Term	Course
Rochester Institute of Technology	Spring 2012: 1 section Fall 2014: 2 sections Spring 2014: 1 section	Programming Language Concepts
Howard University	Spring 2014: 1 section	CSIII
SUNY at Oswego	Spring 2013: 1 section Spring 2015: 1 section	Web services

As we had limited time for teaching SOP, most of the course modules took one week, including one or two lectures and one lab session. Students learned SOP through different channels that included lecture slides, class discussion, programming assignment, and exams. Students were also pointed to a website where they could find the demo, source code, and documentation of various web services and SOP applications.

Assessment and Evaluation

We assessed the proposed approach to incorporating new coursework into computer science in the following aspects: (1) the deployment of the course modules in CS courses, (2) the improvement of cognitive behavior of students, and (3) the improvement of student affective behavior.

As we described in Section 4, both course modules were deployed at each of the three institutions starting from 2012, which was soon after the modules were developed. They were taught in 12 class sessions. The total number of students that have learned SOP through the course modules is 310, including 149 at the first institution, 72 at the second institution, and 89 at the third institution. Considering that it may take years of time and effort to add a new course to an existing curriculum, the use of course modules represents an effective approach to introducing new concepts to students in a timely manner. Moreover, an entry-level or mid-level programming class, such as CS2 or Programming Language Concepts, is usually a required core course in a typical CS curriculum, or has a large size of audience. Incorporating a course module into such a course can expose much more students to new ideas and concepts than introducing a new course.

For student cognitive behavior improvement, we focus on evaluating if the developed teaching materials improve student SOP knowledge. Students are expected to be able to explain the fundamental concepts, such as service-oriented architecture, web service standards, and the principle of SOP methodology after taking the class. Students are also expected to gain the SOP programming skills from the lab sessions, such as building applications using public or private web services, wrapping an existing application as a web service, setting up and configuring web servers to host web services. Therefore, we designed different quiz/exam question libraries and programming assignments for CM1 and CM2 based on their coverage and expected student learning outcomes. Instructors of the classes selected the questions from the libraries and decided the format of them (e.g., multiple choice or short answer question) in the exam. We collected and aggregated the scores from the three institutions. The result shows that around 56% students scored 70\% or better for CM1 and around 84\% students scored 70% or better for CM2. It is worth to note that, some instructors selected two questions in multiple choice question (MCQ) format. In this case, a student scoring 70% or above means that the student actually scored 100%. In particular, students scored much better in their programming assignments, especially after more detailed instructions, examples, and sample codes were provided. From the assessment result, we can see that the majority of students had gained a good knowledge of SOP and the related programming skills.

For student affective behavior improvement, we focus on evaluating if the course modules have inspired students' interests in learning SOP, increased their confidence in their SOP skills, and improved their interests in applying SOP in further endeavors. Therefore, we designed a survey form for each course module and collected students' feedbacks from all three institutions after they took the course modules. Examples of survey questions for CM1 include: *I am able to develop a simple SOP application during the course*, *I am looking forward to taking other courses in SOP and related areas*, and *The SOP module increased my interest in computing or programming*. Examples of survey questions for CM2 include: *I feel confident that I am acquiring the skills to develop a SOP application*, *I am looking forward to using a SOP component in other projects or courses*, and *I possess sufficient fundamental skills to apply for a SOP-related job*. Typical five-level Likert scale responses were designed, ranging from *strongly disagree* (i.e., 1) to *strongly agree* (i.e., 5). For CM1, the calculated averages of the ranged from 2.5 to 3.4. Although the result didn't meet our expectation, it gave us valuable lessons on improving the teaching material and the way of teaching it. We will discuss the lesson in Section 6. For CM2, the calculated averages of the responses were all above 3.1 and most of them fell in the range of 3.6 to 4, which is encouraging but still can be improved.

Lessons Learned

We had learned several lessons from the experience of deploying the SOP course modules. Due to the space limit, in this paper we choose to present the two most important ones, which are summarized as follows.

Lab Setup: Hands-on exercises take a heavy weight on learning SOP. Therefore, a lab environment needs to be set up for students to practice what they learned from the class. Developing and hosting web services request multiple platforms, including a Java SDE, an application server, and a web server. It involves significant effort to install, configure, and integrate those components together. The software package may also be incompatible with or limited by the existing lab hardware and software systems. As SOP is a new technology, compared to traditional programming technologies, it does not have many resources or supports, such as documentations or online forums, to seek for when an issue is encountered. This adds to the difficulties to lab setup and software configuration. Unexpected errors caused by configuration issues during a lab session could seriously ruin student interests in the topic and their learning experience. Software configuration issues are not unique to SOP. We would face those issues when teaching many other new technologies in computer science. To ensure a smooth delivery of the course module, software configuration issues need to be adequately addressed and the lab environment needs to be thoroughly tested beforehand.

Course Module Adoption: We have made significant efforts on promoting the course modules and convincing instructors to adopt them in their classes, including a series of conversations with the instructors , reaching out to local colleges, and hosting regional workshops or the ones at major CS education conferences (e.g., FIE and SIGCSE) to present the course modules and train the instructors, and so on. We received very positive feedback from the instructors regarding the content of SOP course modules.

However, we still face several challenges to get our course modules adopted by more class sections and the instructors in different universities. First, although many instructors showed their interests in SOP and acknowledged that students could benefit from the course modules, their lack of SOP background and the potential considerable time investment on learning the topics made them hesitate to teach them right away. Student learning experience and performance can also be suffered by an instructor's insufficient SOP expertise, which was indicated by the collected assessment data.

Second, for a low-level programming class, such as CS2 and PLC, the teaching schedule is usually already designed to cover its topics. This means a limited buffer, especially in terms of lecture times, for adding new topics. This adds uncertainty to the deployment of course modules even after an instructor plans to teach SOP in his or her class.

Third, for a large CS program, it is common that multiple sections of a programming course, are offered in the same term. Maintaining consistency among different sections requires support from all instructors to incorporate our course modules into that course. Meanwhile, if the instructors do not have enough SOP background, it is difficult, especial in terms of scheduling, to arrange a guest lecturer to teach all the sections. To tackle these challenges, possible solutions could be: (1) start the conversation with the instructors as early as possible; even before the course material is fully developed. This will help reserve lecture time and give instructors more time for making arrangement. (2) Continue our efforts to train the programming course instructors the SOP knowledge so they can be

more convinced and motivated to teach them. This will also solve the guest lecturer issue. (3) Design the lightweight and more detailed version for each course module so that it will take less lecture time and leave some work for student to do after class.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Awards DUE-1140567, DUE-1141112, and DUE-1141200. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. The authors also thank the anonymous reviewers for their feedback.

Reference:

1. G. Bieber, L. Architect, and I. Ci. Introduction to service-oriented programming. In Openwings, 2001. http://www.openwings.org.
2. P. Brusilovsky. Webex: Learning from examples in a programming course.In WebNet, pages 124-129, 2001.
3. R. Burow and G. Weber. Example explanation in learning environments. In Proceedings of the Third International Conference on Intelligent Tutoring Systems, pages 457-465, London, UK, 1996. Springer-Verlag.
4. M. N. Huhns and M. P. Singh. Service-oriented computing: Key concepts and principles. IEEE Internet Computing, 9:75-81, January 2005.
5. E. Knorr. Software as a service: The next big thing. http://www.infoworld.com/d/applications/software-service-next-big-thing-319 , March 2006.
6. B. B. L. Lim, C. Jong, and P. Mahatanankoon. On integrating web services from the ground up into cs1/cs2. SIGCSE Bull., 37(1):241-245, Feb. 2005.
7. A. Sillitti, T. Vernazza, and G. Succi. Service oriented programming: A new paradigm of software reuse. In Software Reuse: Methods, Techniques, and Tools, volume LNCS 2319, pages 39-47, 2002.
8. K. Sung, C. Hillyard, R. L. Angotti, M. W. Panitz, D. S. Goldstein, and J. Nordlinger. Game-themed programming assignment modules: A path-way for gradual integration of gaming context into existing introductory programming courses. IEEE Transactions on Education, 2010.
9. Q. Yu, X. Liu, A. Bouguettaya, and B. Medjahed. Deploying and managing web services: issues, solutions, and directions. The VLDB Journal, 17:537-572, May 2008.