Service-Oriented Computing and Software Integration in Computing Curriculum

Yinong Chen¹ and Zhizheng Zhou²

¹School of Computing, Informatics, and Decision Systems Engineering, Arizona State University
Tempe, AZ 85287-8809, U.S.A.
²School of Computer Science and Technology, Shandong University of Finance and Economics
Shandong, China

Abstract — Web software development and cloud computing based on Service-Oriented Architecture (SOA) represent the latest parallel distributed computing theories, practices, and technologies. As a distributed software development diagram, SOA is being taught in many computer science programs. We do not suggest using SOA to replace the currently taught Object-Oriented Computing paradigm. As SOA is based on Object-Oriented Computing, we suggest teaching SOA as the continuation and extension. At Arizona State University, SOA paradigm is incorporated into our Computing Science program since 2006. This paper presents the topics of the related courses and the open resources created for the courses, which are available for public accesses, including textbooks, lecture presentation slides, tests and assignments, software tools, and a repository of components, services and applications.

Keywords – Service-oriented architecture, Robot as a Service, software integration, service repository, computing curriculum

I. INTRODUCTION

Software development has evolved for several generations from imperative, procedural, object-oriented, to distributed object-oriented computing paradigms [1][2]. As the emergence of Service-Oriented Architecture (SOA) and Service-Oriented Computing (SOC) [3][4][5], software development is shifting from distributed object-oriented development, represented by for example, CORBA [5] developed by OMG (Object Management Group) and Distributed Component Object Model (DCOM) [7] developed by Microsoft, to service-oriented development (SOD). SOA, SOC, and SOD have been adopted and supported by all major computer companies. The related technologies have been standardized by OASIS, W3C, and ISO [8]. Government agencies also adopted SOC in their system development [9].

Before proceeding further, we present the fundamental concepts used in the paper and in other literatures: SOA, SOC, SOD, and cloud computing [3][4][5][8].

A system is typically described by three aspects: architecture, interface, and behavior that defines the transformation of input to output. SOA addresses the architecture aspect, which considers a software system consisting of a collection of loosely coupled services that communicate with each other through standard interfaces and standard protocols [3][4]. These services are platform independent. Services can be published in public or private directories or repositories for software developers to compose their applications. As a software architecture, SOA is a conceptual model that concerns the organization and interfacing among the software components (services). It does not concern the development of operational software.

SOC on the other hand addresses the interface and behavior aspects of a system. SOC includes service interface such as WSDL and communication protocols such as SOAP and HTTP, computing concepts and principles, methods, algorithms, and coding.

SOD concerns the entire software development life cycle based on SOA and SOC, including requirement, specification, architecture design, composition, service discovery, service implementation, testing, evaluation, deployment, and maintenance in a given environment [8]. SOD also involves using the current technologies and tools to effectively produce operational software.

In the Computer Science program at the Arizona State University (ASU), particularly in the Software Engineering concentration, SOA, SOC, and SOD are taught in a number of courses, including the freshman course CSE101/FSE100 (Introduction to Engineering) [10], the sophomore course CSE240 (Introduction to Programming Languages) [2], senior course CSE445 (Distributed Software Development), and CSE446 (Software Integration and Engineering).

This paper will discuss CSE101/FSE100, CSE445 and CSE446, where parallel and distributed computing is the key topics of these courses. While teaching these courses, we collected and created abundance of resources, which are made available to all instructors and students around the world, including textbooks [2][8][10], syllabi [11][18][19], presentation slides, tests and assignments, a repository of sample services and applications for public accesses [23]. We will also present the outreach of the course in our partner universities in China.

In the rest of the paper, sections II presents the service-oriented robotics programming in CSE101/FSE100. Sections III and IV discuss the core topics of SOA, SOC, and SOD taught in CSE445 and CSE446.
Section V presents the repository of resources. Section VI outlines the textbook developed for the courses. Section VI shows, as an example, the enrollments history and evaluation scores of CSE 445/598.

II. SERVICE-ORIENTED ROBOTICS PROGRAMMING

CSE101 is a course required for all students in Computer Science program and Computer System Engineering program at Arizona State University. The course consists of one lecture hour and three lab hours each week for 15 weeks. The main tasks in the three lab hours are using service-oriented robotics development to learn the engineering design process. The development environment used is Microsoft Robotics Developer Studio (MRDS) and its Visual Programming Language (VPL) [12] and Web-based robotics programming environment developed at Arizona State University [17]. First released in 2006, MRDS and VPL laid an important milestone in service-oriented computing. VPL is architecture-driven and is a service-oriented programming language that allows students to develop services, deploy the services into a repository, and then use the services in the repository to develop workflow-based robotics applications.

MRDS is based on the .Net framework. VPL, C#, and Visual Basic can be used to program services as well as applications. MRDS support distributed and event-driven programming model. The services can be added into MRDS’s service repository. As a contribution to MRDS, ASU Repository has included a number services and applications that can be added into MRDS repository and made MRDS services.

We piloted the first offer of the service-oriented robotics development in CSE10 in Fall 2006. The course evolved to the form of today after eight years’ offerings. A number of resources have been developed to support the course, including syllabus [11], the lab manual [12], videos of sample robotics projects [11], a repository of services and sample applications [23]. Figure 1 shows the Web-based programming environment using the concept of Robot as a Service. As the services hide the hardware and programming details, it allows students to better understand different maze algorithms [21]. Using this simple Web environment, student can design an autonomous maze navigation algorithm, such as a short-distance-based greedy algorithm and a wall-following algorithm without prior programming knowledge. A maze navigation program can be written using a few drop-down commands. The virtual robot in the Web can communicate and synchronize with the physical robot to add excitement to the learners. After students have understood the algorithm, they can use VPL to write the program that can run in MSRD simulation environment and using physical NXT robot.

As the materials are easy to learn, exciting, and educational, we have proposed to teach the service-oriented robotics development as a part of the high school computing course. The proposal was funded by the U.S. Department of Education’s FIPSE program (2007 to 2010) [13], which leads to the implementation of the course in Coronado High School [14][15] and the summer Robotics Camps for high school students [16]. A teacher’s training camp has been established using the funding from FIPSE and from Arizona Science Foundation. Over 60 teachers has been trained since 2007, and the service-oriented robotics programming course has been disseminated in many high schools in Arizona through these teachers.

III. DISTRIBUTED SOFTWARE DEVELOPMENT

In the Computer Science (CS) program at ASU, there is a Software Engineering (SE) concentration. SE Students are required to take the following four courses, in addition to the required CS core courses:

- CSE445: Distributed Software Development
- CSE446: Software Integration and Engineering
• CSE460: Software Analysis and Design
• CSE464: Software Quality Assurance and Testing

CSE445 and CSE446 form a sequence and are designed to teach the latest distributed computing and software integration techniques in service-oriented computing, as well as the latest technologies supporting the development of service-oriented software. The topics of CSE445 will be discussed in this section and the topics of CSE446 will be presented in the next section.

2.1 Course Objectives, Outcomes and Contents
CSE445 is the first class in our computer science program to discuss parallel and distributed software development in depth. The objectives and outcomes of a course include [18]:

1. To develop an understanding of the software engineering of programs using concurrency and synchronization, with the following outcomes: Students can identify the application, advantages, and disadvantages of concurrency, threads, and synchronization; Students can apply design principles for concurrency and synchronization; Students can design and write programs demonstrating the use of concurrency, threads, and synchronization.

2. To develop an understanding of the development of distributed software, with the following outcomes: Students can recognize alternative distributed computing paradigms and technologies; Students can identify the phases and deliverables of the software lifecycle in the development of distributed software; Students can create the required deliverables in the development of distributed software in each phase of a software lifecycle; Students understand the security and reliability attributes of distributed applications.

3. To develop an ability to design and publish services as building blocks of service-oriented applications, with the following outcomes: Students understand the role of service publication and service directories; Students can identify available services in service registries; Students can design services in a programming language and publish services for the public to use.

4. To build skills in using a current technology for developing distributed systems and applications, with the following outcomes: Students can develop distributed programs using the current technology and standards; Students can use the current framework to develop programs and Web applications using graphical user interfaces, remote services, and workflow.

CSE445 is designed to achieve the objectives and outcomes of the course. The course contents are organized in six units:

1. Introduction to Parallel and Distributed Computing: This unit gives an overview of the area and the main topics to be discussed in the course. It covers parallel and distributed computing paradigm, distributed software architecture, design patterns, concepts of service-oriented architecture, service-oriented computing, and service-oriented software development.

2. Parallel and Distributed Computing with Multithreading: This unit covers the fundamental issues in parallel distributed computing, including critical operations, synchronization, resource locking versus unbreakable operations, semaphore, events and event coordination, and event-driven distributed computing in Web environment. As a part of our parallel computing initiative [24], we also cover the performance issues of multithreading and distributed computing under the multi-core architecture support [25].

3. Essentials in Service-Oriented distributed Software Development: This unit covers the essential components and gives a quick start of developing distributed software, including development environments, service-oriented computing standards and interfaces, developing services as service providers, understanding service brokers, discovering and publishing services in service brokers, and developing service clients consuming services.

4. XML Data Representation and Processing: This unit discusses XML and related technologies supporting including XML fundamentals, XML data processing in SAX, DOM, and XPath models, XML type definition and schema, XML validation, and XML Stylesheet language.

5. Web Application and Web Data Management: This unit elaborates the development of Web applications distributed in different service providers. It covers the models of Web applications, structure of Web applications, state management in Web applications. Database and caching support to Web application state management are discussed in the next course, as a tradeoff between the materials in the sequence of two courses. This unit also covers dynamic graphics generation to leverage the presentation of Web applications at the programming level.

6. Dependability of Web Software: Dependability, including reliability and security, is a more important issue in Web applications than that in desktop applications. In addition to present essential issues in dependability design of Web applications, this unit also designs and implements the security mechanisms that safeguard the Web applications.

2.2 Sample Projects
In each of the main topics, a project is given. In this section we present the project for two of the topics.

In the multithreading part of the course, Intel’s Thread Building Blocks (TBB) are discussed in class for server application design. TBB provides a straightforward way of introducing performance
threading, by turning synchronous calls into asynchronous calls and converting large methods (threads) into smaller ones. To demonstrate the performance impact, a program that validates the Collatz conjecture has been used to evaluate the performance in a single core up through 32 cores using Intel Manycore Testing Lab (MTL). Figure 3 shows the measured speed-up and the usage efficiency for 4, 8, 16, and 32 cores with respect to a single core.

![Figure 3. Speedup and efficiency](image)

In the final project, students will develop a working Web application and deploy the application into a Web server for public access. Figure 4 shows a sample project, where students must develop both the client and the provider. From the client side, an end user applies for an account by submitting necessary information. The provider issues a user ID if the application is approved. Using the ID, the end user can create password and then access the system.

![Figure 4. Web application project](image)

The project involves GUI design at the presentation layer, programming at business logic layer, and data manipulation and storage at data management.

### 2.3 ACM CS Curriculum Compliance

All ASU classes are designed based on ACM CS curriculum. This course covers the ACM CS topics listed in Tables 1, 2, and 3, which relate the topics to the Learning Objectives in Bloom's Taxonomy and their learning outcomes in programming, algorithm, and cross cutting and advanced topics, respectively.

Currently, many of the topics we teach in this course are listed in cross the category of cutting and advanced topics. We believe these topics will be considered standard contents in near future.

In the Tables 1 through 3, the Bloom's Taxonomy abbreviations are: Knowledge (K), Comprehension (C), and Application (A).

#### Table 1. ACM CS Programming topics

<table>
<thead>
<tr>
<th>Topics</th>
<th>Bloom #</th>
<th>Learning Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client Server</td>
<td>C</td>
<td>Know notions of invoking and providing services (e.g., RPC, RMI, web services) - understand these as concurrent processes.</td>
</tr>
<tr>
<td>Task/thread spawning</td>
<td>A</td>
<td>Be able to write correct programs with threads, synchronize (fork-join, producer/consumer, etc.), use dynamic threads (in number and possibly recursively) thread creation - (e.g. pthreads, CILK, JAVA threads, etc.).</td>
</tr>
<tr>
<td>Libraries</td>
<td>A</td>
<td>Know one in detail, and know of the existence of some other example libraries such as Pthreads, Pfunc, Intel's TBB (Thread building blocks), Microsoft's TPL (Task Parallel Library), etc.</td>
</tr>
<tr>
<td>Tasks and threads</td>
<td>K</td>
<td>Know the relationship between number of tasks/threads/processes and processors/cores for performance and impact of context switching on performance</td>
</tr>
<tr>
<td>Synchronization</td>
<td>A</td>
<td>Be able to write shared memory programs with critical regions, producer-consumer, and get speedup; know the notions of mechanism for concurrency (monitors, semaphores, … - [from ACM 2008])</td>
</tr>
<tr>
<td>Performance metrics</td>
<td>C</td>
<td>Know the basic definitions of performance metrics (speedup, efficiency, work, cost), Amdahl’s law; know the notions of scalability</td>
</tr>
</tbody>
</table>

#### Table 2. Algorithms topics

<table>
<thead>
<tr>
<th>Topics</th>
<th>Bloom #</th>
<th>Learning Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speedup</td>
<td>C</td>
<td>Use parallelism either to solve same problem faster or to solve larger problem in same time</td>
</tr>
<tr>
<td>Scalability in algorithms and architectures</td>
<td>K</td>
<td>Understand that more processors does not always mean faster execution, e.g. inherent sequentiality of algorithmic structure, DAG representation with a sequential spine</td>
</tr>
<tr>
<td>Dependencies</td>
<td>K, A</td>
<td>Understand the impact of dependencies and be able to define data dependencies in Web caching applications</td>
</tr>
</tbody>
</table>
Table 3. Cross cutting and advanced topics

<table>
<thead>
<tr>
<th>Topics</th>
<th>Bloom #</th>
<th>Learning Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cloud</td>
<td>K</td>
<td>Know that both are shared distributed resources - cloud is distinguished by on-demand, virtualized, service-oriented software and hardware resources</td>
</tr>
<tr>
<td>P2P</td>
<td>K</td>
<td>Server and client roles of nodes with distributed data</td>
</tr>
<tr>
<td>Security in Distributed System</td>
<td>K</td>
<td>Know that distributed systems are more vulnerable to privacy and security threats; distributed attacks modes; inherent tension between privacy and security</td>
</tr>
<tr>
<td>Web services</td>
<td>A</td>
<td>Be able to develop Web services and service clients to invoke services</td>
</tr>
</tbody>
</table>

As the domain is new, no existing textbooks adequately cover these materials. There are books that cover the concepts and principles well, but they do not discuss the development of operational software. There are books that focus on service-oriented software development. Those books are mostly platform-based and do not associate the concepts and principles to the software in development. We have developed a textbook to facilitate this course, which consists of detailed explanation of concepts, using examples to illustrate each key concept, and using case studies to link multiple concepts together and to provide working examples. Part I of the text [8] is dedicated to teach this course. Assignments and projects are given at the end of each chapter.

Students taking CSE445 are expected to have good programming background in an object-oriented programming language such as C++, Java, or C#, and basic software engineering background. All development tasks are language-based either in Java or in C#, including multithreading software development, Web service development, and Web application development. In contrast, the next course, CSE446, will focus more on the software composition and integration using existing services and components.

**Evaluation plan:** A number of assignments are assigned. In the multithreading programming assignment, students will test their program in both single core and multicore environments. Students will also revise the program using Intel’s TBB library and measure the speed-up. In the service hosting assignment, students will explore parallelism on the server side and determine the performance improvement based on the service model and the multicore efficiency.

IV. SOFTWARE INTEGRATION AND ENGINEERING

CSE446 (Software Integration and Engineering) course is built on the basic concepts and principles discussed in CSE445, yet they do not rely on the detail of CSE445. If both courses are offered, CSE445 should be offered before CSE446. However, with a review of the basic concepts and preparation in XML, this course can be taught independent of CSE445. CSE446 emphasizes software composition and integration using existing services and components. The approach is based on higher-level of data management and application building techniques. The objectives and outcomes of the course are [19]:

1. To understand software architecture and software process, with outcomes: Students understand the requirement and specification process in problem solving; Students understand software life cycle and process management; Students can identify advantages and disadvantages of software architectures and their trade-offs in different applications.

2. To understand and apply composition approach in software development: Students can apply software architecture to guide software development in the problem solving process; Students understand interface requirement of software services; Students can compose software based on interfaces of services and components; Students can develop software system using different composition methods and tools.

3. To understand and apply data and information integration in software development: Students can compose software systems using different data resources in different data formats; Students can integrate application logic with different databases; Students can apply the entire software life cycle to develop working software systems. CSE446 is designed to achieve the objectives and outcomes of course. The course materials are organized in seven units.

1. Distributed Application Architecture
2. Advanced Architecture-Driven Application Development
3. Enterprise Software Development and Integration
4. Event-Driven Architecture and Applications
5. Interfacing Service-Oriented Software with Databases and Big Data analysis
6. Ontology and Semantic Web
7. Cloud Computing and Software as a Service

Cloud computing is a key unit in the course, as it is the latest technology that all the major computing companies and governments are supporting and utilizing. In the U.S. Federal Cloud Computing Strategy, published in 2011, Vivek Kundra, the U.S. Chief Information Officer, planned to spend $20 billion of the entire government’s $80 billion IT budget in cloud computing [22].

Another highlight of the course is of teaching the workflow-based software development, which turns the dream of generating executable directly from the
The purpose of SOC is to improve the quality and productivity of software and application development. It can succeed only if a large repository of services is available. Ideally, all services developed by all corporations and by individuals will be open to the public in public directories and repositories. The current situations are:

- **Private Services**: Many corporations, for example, IBM and SAP, keep their repositories private for internal use only. These services will not be available for education purposes.

- **Paid Services**: Many corporations, for example, Amazon Web Services, offer commercial services and subscription and payments are required. It is obviously correct and necessary for the application holders to pay for the services they use, in order to reflect the value of the services and the entire service-oriented computing paradigm, as well as maintain the service agreement between the service providers and service clients. However, such services are not useful for education purposes, as we cannot ask students to use these services to build their course projects, although many of our students paid for the services in order to develop better assignment projects.

- **Free public services**: There are a number of service directories where free public services are listed, including Xmethods.net, Webservicex.net, and remotemethods.com. Google and Microsoft offer free services and APIs in a number of areas in search and map services. Free public services are great resources for education purposes, which are the main sources our students use to develop their applications. There are several problems with the free public services.

  - The number of services and the range of services are limited.
  - The performance of some of the services is not adequate. The services are too slow to use (frequent timeout when many students are accessing). The situation occurs particularly before an assignment is due and a large number of students are accessing the free services.
  - The availability, reliability, and maintainability are not warranted. Services are often offline or be removed without notice. Service interfaces and implementations can be modified too.

To reduce the possible problems, we have developed a repository of our own at:

http://venus.eas.asu.edu/WSRepository/

This repository complements the free public services in several ways. We develop services according the need of the course and its assignments. The source code of the services and applications are open and explained as examples in the text [8]. The services and applications include simple function services that illustrate the development process, for example, encryption and decryption services, access control services, random number guessing game services, random string (strong password) generation services, dynamic image generation services, random string image (image verifier) service, caching services, shopping cart services, messaging buffer services, and mortgage application/approval services. The services are implemented in multiple formats, including ASP.Net services, Windows Communication Foundation services, RESTful services, and Work Flow services. All the services are free and open to the public. We maintain the server to keep the high availability and reliability of the services.

We also developed a service directory that lists services offered by other service directories and repositories using a service crawler that discovers available services online. The service directory can be accessed through a service engine at: http://venus.eas.asu.edu/sse/. We also offered a registration page for anyone to list their services into the service directory. The registration is at:

http://venus.eas.asu.edu/sse/ServiceRegister.aspx

### VI. The Textbook

As we discussed in the previous sections, a textbook is necessary to support these new courses and is developed that summarizes the main contents of the four courses in SOA, SOC, SOD, and cloud computing [8]. Three editions have been published, and the fourth edition of the book to be printed in fall 2014 has the following chapters, which are organized in three parts for the three courses.
Part I  Distributed Service-Oriented Software Development and Web Data Management
Chapter 1  Introduction to Distributed Service-Oriented Computing
Chapter 2  Distributed Computing with Multithreading
Chapter 3  Essentials in Service-Oriented Software Development
Chapter 4  XML Data Representation and Processing
Chapter 5  Web Application and State Management
Chapter 6  Dependability of Service-Oriented Software

Part II  Advanced Service-Oriented Computing and System Integration
Chapter 7  Advanced Services and Architecture-Driven Application Development
Chapter 8  Enterprise Software Development and Integration
Chapter 9  Internet of Things and Robot as a Service
Chapter 10  Interfacing Service-Oriented Software with Databases
Chapter 11  Big Data Systems and Ontology
Chapter 12  Service-Oriented Application Architecture
Chapter 13  A Mini Walkthrough of Service-Oriented Software Development
Chapter 14  Cloud Computing and Software as a Service

Part I of the text is used for CSE445. Part II is used for CSE446. The book has been translated into Chinese and is used in multiple universities in China [8].

VII. Enrollment and Evaluation

CSE445 was redesigned in Fall 2006 to reflect the new development paradigm in distributed software development. CSE445 was named Distributed Computing with Java and CORBA. The course was renamed to Distributed Software Development and the contents are changed to base on service-oriented computing, as discussed in the previous section. CSE445 is required for Software Engineering concentration, which is a part of CS program. There are about 50 students are enrolled in SE concentration. CS and CSE students can take CSE445 as a credit elective. CSE445 is also listed as CSE598 as an elective of graduate students. The enrollment numbers of CSE445/598 are given in Table 4.

Figure 5 plots the data, which visualize the enrollment in of CSE445 and CSE598. Both sections show significant increases from 2006 to 2014. The combined enrollment has increased from 39 in Fall 2006 to 134 in Fall 2013.

Course evaluation is done at the end of the course by all students. Table 5 shows the average scores of CSE445/598 (Distributed Software Development) since Fall 2006. The scores are out of 5.0, where 5.0 is very good, 4.0 is good, 3.0 is fair, and 2.0 is poor. Students are and excited of learning the latest computing theories and practice before they enter the job market.

<table>
<thead>
<tr>
<th>Year</th>
<th>Semester</th>
<th>445 enrollment</th>
<th>598 enrollment</th>
<th>Enrollment total</th>
</tr>
</thead>
<tbody>
<tr>
<td>2006</td>
<td>Fall</td>
<td>25</td>
<td>14</td>
<td>39</td>
</tr>
<tr>
<td>2007</td>
<td>Spring</td>
<td>16</td>
<td>16</td>
<td>32</td>
</tr>
<tr>
<td>2007</td>
<td>Fall</td>
<td>24</td>
<td>21</td>
<td>45</td>
</tr>
<tr>
<td>2008</td>
<td>Spring</td>
<td>39</td>
<td>8</td>
<td>47</td>
</tr>
<tr>
<td>2008</td>
<td>Fall</td>
<td>35</td>
<td>23</td>
<td>58</td>
</tr>
<tr>
<td>2009</td>
<td>Spring</td>
<td>38</td>
<td>13</td>
<td>51</td>
</tr>
<tr>
<td>2009</td>
<td>Fall</td>
<td>33</td>
<td>10</td>
<td>45</td>
</tr>
<tr>
<td>2010</td>
<td>Spring</td>
<td>38</td>
<td>22</td>
<td>60</td>
</tr>
<tr>
<td>2010</td>
<td>Fall</td>
<td>42</td>
<td>34</td>
<td>76</td>
</tr>
<tr>
<td>2011</td>
<td>Spring</td>
<td>50</td>
<td>20</td>
<td>70</td>
</tr>
<tr>
<td>2011</td>
<td>Fall</td>
<td>30</td>
<td>52</td>
<td>82</td>
</tr>
<tr>
<td>2012</td>
<td>Spring</td>
<td>52</td>
<td>15</td>
<td>67</td>
</tr>
<tr>
<td>2012</td>
<td>Fall</td>
<td>42</td>
<td>35</td>
<td>77</td>
</tr>
<tr>
<td>2013</td>
<td>Spring</td>
<td>55</td>
<td>38</td>
<td>93</td>
</tr>
<tr>
<td>2013</td>
<td>Fall</td>
<td>44</td>
<td>90</td>
<td>134</td>
</tr>
<tr>
<td>2014</td>
<td>Spring</td>
<td>50</td>
<td>62</td>
<td>112</td>
</tr>
</tbody>
</table>

Based on the feedbacks from students, a follow up class of CSE445/598 class has been designed and pilot-taught in summer 2010 and summer 2011, with 20 students enrolled in both sections. The new course CSE446/598 has been approved as a required course for the Software Engineering Concentration from Spring 2012, which has the same status as CSE445/598. As CSE445 is the prerequisite of
CSE446, the enrollment number of CSE446/598 is slightly lower than CSE445/598

VIII. CONCLUSIONS

The paper presented three parallel and distributed computing courses in the CS curriculum at the Arizona State University. CSE101 and CSE445 have been included in the curriculum since 2006, while CSE446 was added in 2011. For all the major concepts and topics in the courses, working examples are given to make sure that students not only understand the concepts and principles, but also how to implement the concepts and principles in operational software. Application deployment into a Web server is emphasized in CSE445 and CSE446, as it is not covered in other software development courses. The increase in the enrollment of the courses as well as the feedbacks from students show that these courses improved our curriculum and are well received by students. Resources are created and made available for other universities to adopt.

REFERENCES