The Teaching of Net-Centric Computing

David Lefkovitz [1]
CIS Department, Temple University
Philadelphia, PA 19122

Abstract

This paper presents the subject of net-centric computing as one that spans a spectrum from static html web pages to the development and control of distributed, multi-tier components. It poses the dilemma of attempting to teach this important body of knowledge that requires multiple languages and tools in an already tightly packed Computer Science program, and presents a solution that utilizes three pedagogic devices: (1) A series of lab exercises that incrementally spans the defined spectrum and that presents Starters that act as models for the languages being learned, (2) Web based notes and classroom demonstrations, and (3) Online tutorials, reference manuals and white papers. The paper also discusses the problems inherent in the teaching of methodology vs. specific languages, software systems and tools and how it is being approached by expansion of Device 2 into Course Technology Modules.

Keywords: Web-deployed applications, client/server processing, distributed components, multi-tier architecture, middleware

1. INTRODUCTION

Most CS and IS departments are introducing web related courses or course components into their curricula. These fall onto a spectrum that spans (1) static web page (HTML) design, (2) client side processing via scripting, (3) server side processing via ASP, DHTML, XML, cgi processes and servlets, (4) distributed, n-tier components with VB, Java, or other languages using middleware control via DCOM (Distributed Component Object Model), CORBA (Common Object Request Broker Architecture) or EJB (Enterprise Java Beans). Superimposed upon these technologies are conventional file and database management systems. Collectively, we might call these course elements *net-centric computing*.

The problem facing undergraduate programs, and to some degree graduate programs, is how best to teach courses across this spectrum. How many courses should it comprise? Should any of them be required core courses, or should existing ones be modified to accommodate the one or more net-centric courses to follow? At what level should these courses be introduced, and thus what is the required pre-requisite knowledge from traditional core courses like programming, (usually C++ and/or Java), networks, operating systems, and database management? The CS curricula at most Universities are so crowded now that few major electives can be managed, and the required core courses have no room to include much new material that might provide some of the desired pre-requisite knowledge.

1. lefkovitz@cis.temple.edu

Further compounding the problem is the fact that net-centric computing requires several languages and tools that all students may not know beforehand, and finally there is a question of objective and emphasis. Ideally, we strive in Computer Science to teach principles and methodology over the details and nuances of specific languages and tools; however, in net-centric computing one cannot go far without actually implementing some small systems, which means that the student must learn a working subset of such languages as HTML, ASP, XML, VB, Java, and PERL plus associated tools. Also, some curricula do not require database management in the core, which means that SQL must also be learned. Greenspun at MIT has also addressed this problem and has developed a course titled *Software Engineering for Web Applications*. (Greenspun 2000)

This paper describes the steps being taken at Temple University to address these issues. We have started with a one semester, undergraduate elective course originally named *Distributed Component Development* and now changed to *Net-Centric Computing*, CIS 309. The goal was to cover the four point spectrum with a specific selection of languages and tools, where the outcome was an ability of the student to implement projects exhibiting characteristics across the entire spectrum. In addition, the course was to include as much of the principles and methodology as time would allow.

Specific languages used were HTML, ASP, VB and VB script, Jscript (as a student elected option), XML, and SQL. These were taught through a combination of lectures, examples and self teaching, as will be described. Tools, software systems and middleware used were: (1) VB IDE (Integrated Development Environment) for forms and component object development, (2) Visual InterDev for ASP page development, (3) SQL Server 7 DBMS and the SQL Server Enterprise Management Console, and (4) DCOM and Microsoft Transaction Server (MTS) as middleware control of component deployment. The challenge was how to teach all of this material in one course and achieve the desired outcome in the face of having in place only the traditional pre-requisite structure of programming, networks and operating systems. In our case, even database management is an elective.

2. APPROACH

The solution to these problems was approached through the use of three pedagogic devices.

Device 1: Language teaching by example
Ten lab projects were presented that incrementally spanned the four-point Net-Centric Computing spectrum. Each lab comprised 2 parts, a *Starter* and an *Assignment*. The *Starter* demonstrated the methodology of the given increment and contained most of the language elements needed in the *Assignment*. The student could run the Starter to see how the elements of that lesson behaved in the given set of environments being taught (wired network vs. internet, workstation vs. remote server, workstation forms vs. browser, text file vs. relational file system, COM vs. DCOM, 1 vs. 2 vs. n-tiers, etc.). It was also a means by which the limited language lectures and the self teaching were made concrete. A book that was chosen as the required text of the course contained many of the Starter exercises on CD plus many others that the student could freely use (Micro Modeling Associates 1999).

Part 2 of each lab was the student *Assignment*. Here the student was required to extend the Starter in a significant way. Table 1 lists the 10 labs in terms of the distributed layers (Presentation, Business, Data Service) and tiers.

Device 2: Web Notes for lectures and student reference
All lectures were presented via web notes with extensive use of diagrams for explaining architectures and high level concepts. The primary objective of the lectures was to teach the underlying principles and methods of the net-centric computing spectrum. The secondary objective was to teach the needed languages and tools. A smart classroom was used, so that the lecturer could project the web notes and demonstrate use of the software tools. Outside of class, students had access to the notes at any time. They could also print them before coming to class and annotate them by hand during the lecture. The notes were developed in course units that could play a future role in other course developments, described in Section 4, Course Technology Modules.

Device 3: On-Line references for self instruction
The textbook was used primarily for theory and methodology, though it did provide rudimentary chapters on ASP and XML. A bibliography of books for learning the various languages was provided, but this would subject the students to considerable expense. Instead, a list of urls provided tutorials, reference manuals and white papers for each language and tool used in the course.

The reader is referred to the course url for further elaboration, where the actual lab exercises, lecture notes, and Online references are to be found. The url is **ww2.cis.temple.edu/cis309.**

3. LESSONS LEARNED

- The course started with 19 students. Nearly all of them knew some HTML but most not ASP, XML, VBScript and SQL, nor the Visual Studio tools used for VB and ASP. None knew DCOM or MTS. Fourteen students finished the course. All but 3 students completed the 10 assignments; the other 3 completed 8 of them. This provided ample evidence that the 3 devices are able to teach these 4 languages, some database theory and the associated tools, in the absence of specific pre-requisite knowledge. The students said that the 10 Starter examples were crucial to their ability to begin coding immediately. Much of the remainder of the specific knowledge to produce workable code came from the OnLine References and from the Help systems of the two major tools, VB Integrated Development Environment and Visual InterDev. The Lectures and Web Notes relating to the languages also helped, but these were more summary in nature.

- The course requires considerable self-instruction and discipline on the part of the student. It should work well with motivated and interested students in an elective course. It may not work so well in a required course.

- This single course still left some gaps that we felt needed to be addressed, if not filled. As stated, we had to be selective in the choice of languages, software systems and tools. For example, the course taught DCOM as middleware, not CORBA or EJB. It emphasized VB for component generation as opposed to Java or some other language like C++, and the course used SQL Server, not Oracle or another DBMS. Were the omissions important? Some argue that only methodology is important, not the specific languages, systems and tools, because the latter can be learned on the job and may change rapidly. Others argue that students should have exposure to both Microsoft and the Java/CORBA/Oracle technologies. As a practical matter, the methodology cannot adequately be understood without actual implementation, and there is insufficient time in one course to teach more than one set of languages, tools and middleware technologies. It may be desirable, though, for a CS curriculum to have the flexibility to offer variant courses on net-centric methodology using different languages, tools and systems. This flexibility is all the more important when considering a mix of undergraduate and graduate courses. This led us to the concept of Course Technology Modules (CTM). These are intended to be fairly self-contained teaching components from the web notes, that can be combined to teach the same net-centric methodology using different languages, tools and systems and coordinated with their own set of Starter labs. The concept is being applied to some other courses.

4. COURSE TECHNOLOGY MODULES

A Course Technology Module is a set of web notes, possibly with associated lab exercises, that teaches a unit within the general area of net-centric computing and associated areas like database theory. A module usually comprises 1 to 4 hours of lecturing. The intent is that the work of one instructor in developing a net-centric course can be utilized by him/herself or another, in the development of other or variant courses by selecting an appropriate set of such modules. The author has, to date, developed 2 courses in this manner, with 4 variants for undergraduate and graduate use. One course is CIS 309, discussed in this paper. The other is an Oracle course. This Fall our graduate software engineering course, CIS580, will be adapted to a combination of Oracle and DCOM. Development of these CTMs is still at an early stage, but the reader can get some idea of them by visiting the course WebNotes at ww2.cis.temple.edu/cis309 and ww2.cis.temple.edu/cis595.

Table 2 presents a projected list of such modules. At present they are organized into a 2 level hierarchic classification, but as the courses and their modules develop, the classification may be deepened and new classes added at each level. The numbers in parentheses following Module Numbers identify courses in which the modules have been used:

1. CIS308: Net-Centric Computing (with Java/CORBA)
2. CIS309: Net-Centric Computing (with VB/DCOM)
3. CIS331: Principles of Database Management
4. CIS350: Software Engineering (with Oracle tools)
5. CIS580: Software Engineering (with DCOM and Oracle)
6. CIS585: E-Commerce Site Development
7. CIS595: Software Engineering (with Oracle and web deployment)

Unused modules are planned for inclusion in courses to be developed over the next two years.

5. REFERENCES

Greenspun, Philip, 2000, *Software Engineering for Web Applications*, http://philip.greenspun.com/teaching/one-term-web.html.

Micro Modeling Associates, 1999, *Microsoft Commerce Solutions Web Technology*, Microsoft Press, 1999. ISBN: 0-7356-0579-3.

Table 1: Ten Graduated Labs for Net-Centric Computing

Lab No.	Tiers	Layer		
		Presentation	**Business**	**Data Service**
1	1	T1: Browser: HTML, VBScript		
2	3	T1: Browser: HTML, VBScript	T2: Server 1: ASP	T3: Server 1: NT File System
3	3	T1: Browser: HTML, VBScript	T2: Server 1: ASP	T3: Server 1: SQL Server
4	2	T1:Workstation: VB Form	T1:Workstation: VB Standard Module	T2: Server 1: SQL Server
5	3	T1:Workstation: VB Form	T2:Workstation: COM Object	T3: Server 1: SQL Server
6	3	T1:Workstation: VB Form	T2: Server 1: DCOM Object	T3: Server 1: SQL Server
7	4	T1:Browser: HTML	T2: Server 1: ASP T3: Server 1: DCOM Object	T4: Server 1: SQL Server
8	4	T1:Browser, HTML	T2: Server 1: ASP T3: Server 2: DCOM object	T4: Server 1: SQL Server
9	5	T1:Browser: HTML with Form	T2: Server 1: ASP T3: Server 1: 1 DCOM object T4: Server 2: 3DCOMobjects	T5: Server 1: SQL Server
10	3	T1: Browser: HTML table, XML, DHTML	T2: Server 1: DHTML download	T3: Server 1: XML file is accessed by NT File system

Table 2: Course Technology Modules

No	Module	No	Module
1	**Oracle Technology**	**3**	**Java**
1.1 (4,5,7)	8i Database Server (RDB)	3.1(1)	Applets, Servlets, Applications, JSP
1.2 (4,5,7)	PL/SQL and SQL*Plus	3.2 (1)	JDBC
1.3 (5)	8i Database Server (ORDB)		
1.4 (4,7)	Developer 2 tier	**4**	**CORBA**
1.5 (7)	Developer 3 tier (web deployed)	4.1 (1)	ORB, IDL, and Name Service
1.6 (4,5,7)	Designer – Basic	4.2 (1)	Oracle Application Server (CORBA)
1.7	Designer – Advanced	4.3 (1)	EJB (Enterprise Java Beans)
1.8 (7)	Oracle Application Server (Cartridges)		
1.9	Discoverer and WebDB (High level Information Retrieval and Web Reporting)	**5**	**Database and Information Retrieval**
1.10	WebDB: DDL and Web Forms	5.1 (All)	SQL – Basic
1.11	Data Warehousing, Data Mining and OLAP	5.2	SQL – Advanced
1.12	ERP (Enterprise Resource Planning)	5.3 (2,5)	XML – Basic
		5.4	XML – Advanced
2	**Microsoft Technology**	5.5	Web Search Engines
2.1 (3)	Access 2000		
2.2 (2,5)	VB – Basic (Form and standard modules)		
2.3 (2,5)	VB - Advanced (Class modules, compts)		
2.4 (2,5,7)	ADO (ActiveX Data Objects)		
2.5 (2)	SQL Server		
2.6 (2,5)	COM, DCOM and MTS		
2.7 (2,5,7)	ASP		
2.8 (2,5)	Visual InterDev		
2.9 (7)	Front Page		
2.10 (6)	E-Commerce Site Development		