During the past five years, many college and university Web systems have changed dramatically: homegrown legacy systems have been replaced with vendor ERP solutions; employee and financial Web applications have been added to the existing base of student systems; and Web traffic has exploded. Unfortunately, many existing two-tier structures are increasingly unable to handle the current loads.

To accommodate the ever-increasing traffic, optimize programming effort, and support portals and similar session-based applications, a number of higher education institutions—including the University of Utah—have embarked on the conversion to a three-tier, server-side Java infrastructure.

The current enthusiasm about Java concerns its server-side programming and the collection of technologies known as Java 2 Enterprise Edition (J2EE). A new, third tier—the *application* server—handles transactions, security, database connection pools, and threads, leaving programmers free to focus on coding business logic in the form of Enterprise Java Beans (EJBs). This delineation of processing, along with the ability to cache and share objects, renders the three-tier environment highly scalable and efficient. In addition, Java's portability has made it the standard platform for smartcards and other mobile computing devices: imagine delivering grades and other student information to hand-held computers or cell phones!

For those colleges and universities that have set the direction to Java, the challenge is to determine the best way to approach the change, given barriers that include workload, staff expertise, funding, and a shift from structured programming to an object-oriented methodology. Institutions must assess a clear idea of your peak usage periods during the year and the size of your customer base for each application. If you haven't done so already, start tracking response time for critical applications during peak days. For example, what is your threshold for response time for adding and dropping classes in your registration system or for viewing grades?

Another area to assess is your project queue. How many new Web applications or enhancements are scheduled, and what are their deadlines? Will any of the new applications drastically expand your customer base or the use patterns of your existing base? If you have large projects in the queue, do you really want to write them once in the current environment and then again in the new? If your Web systems interface with a vendor product on the back end, examine the product release schedules. Is your vendor moving to Java sooner or later than you had planned? How long will your current interface be supported? Is there an impending upgrade that will affect your timing?

Don't forget to plan ahead for the conversion of all production systems. At the University of Utah, we are converting the core Web systems first, then using that experience to estimate the conversion effort for the systems and requests remaining in the queue. Educate your customers about the conversion process, and set a cut-off date for development in the old system. Plan for a time period when both new and old systems will be running concurrently, and identify sys-
Dedicate Resources
The single most important factor in a successful conversion is to have staff whose time is dedicated to the project. Java is an object-oriented language, and for most “legacy” analysts and programmers, thinking in objects takes considerable time, effort, and practice. It is unrealistic to expect your staff to work at 100 percent efficiency while they are mentally straddling two diverse development patterns. Jeremy Lund, Web systems analyst and team lead for the conversion at the University of Utah, notes: “The best way to reinforce the new design is by immersing yourself in it.”

One way to devote your human resources is to hire additional staff solely for the Java conversion. For this option to be successful, not only must you have the budget and good fortune to find experienced Java developers, but your existing systems must be well documented from both a programming and a design perspective so that the new staff can get a quick start. The outsourced or newly hired Java developers must also have an understanding of your environment, of your goals, and of the collective expertise of in-house programming staff.

The option that is probably more feasible for most higher education institutions is to allocate existing staff to the project. The primary advantage of using current team members for the conversion is that they are already familiar with system functionality, customer business rules, and overall department goals and objectives. At the University of Utah, the best way to allocate existing staff was to impose a temporary freeze on Web development, which freed up most of the programmers’ time for the conversion. The freeze was a “soft” freeze in that the university identified the top critical projects or enhancements for each customer and accounted for the time it would take to support those critical projects and production issues.

Plan on dedicating hardware and software resources as well. Set up a separate machine for the conversion project, and start the three-tier structure fresh with the application server install, Java components and interfaces, new directory structures, and naming conventions. Clearly separate old and new.

Examine the Technology’s Scope
The Java programming language is part of the very powerful and complex J2EE platform. Take the time to examine the collection of technologies and start with only those that are needed in your environment. For example, you may need only the database connection (JDBC) and JavaServer Pages (JSP) or servlets to support the same functionality of your existing programs. Model the initial components so that they are easily expandable to include other pieces of the J2EE platform. If you are bringing in consultants or experienced Java developers for your conversion, make sure that they implement technologies at a technical level that can be supported by your application and systems programming staff.

Train and Educate
As with most new technologies, training for Java is imperative. How you go about this training will depend on your budget, time frame, and the skills of current staff. Princeton University has a successful in-house training program, created by Tim Hogan, in which staff are given a list of prerequisite tasks that they must complete before coming to the classroom. Recognizing the different learning methods, the Princeton program offers staff a variety of Web tutorials, computer-based training courses, and books.

This is a good time to reexamine some of the roles of the Web team members. In many existing systems, particularly Perl/CGI-based programs, the page layout and design is embedded in the program code. In the Java environment, design and logic are more easily separated, which means that programmers can focus on logic while Webmasters and interface designers can work on layout and design, which can be changed at any point independently of the underlying business code. Realize that you will be training more than just the programming staff; systems analysts will be learning about object design methods; systems programmers will be installing and maintaining a new tier, the Web application server; staff in quality assurance will be substantially changing code promotion procedures; and Webmasters will be using new tools to enable/disable links and to troubleshoot systems.

Although you can expect some reluctance and trepidation from staff, there will also be a buzz of excitement as they see new opportunities for their skills, both inside and outside of the university. New skills may emerge as legacy programmers experiment with new technologies; observe and match skills to roles. For example, Cobol programmers will likely be comfortable with servlets, whereas data and system analysts will probably grasp EJBs quickly, given their experience with data modeling.

Finally, educate your customers. Get their buy-in by involving them in the process and educating them on the issues. Help them understand the reasons for the conversion.

Seize the Opportunity!
This is a great opportunity to collect all of the “I wish we would have” ideas from the past year or two and turn them into features of the new system. If you have a list of enhancements from your customers, bundle them together now as part of the new system. Be sure your programmers understand the need for documentation, and make this a part of the system requirements. The standards and systems you build now will be the foundation for the next several years. Do it right!

Note
1. If you are interested in more information about Java, start with Sun’s Web site: <http://www.java.sun.com/>.

For Java information specific to higher education, visit the Java in Administration Special Interest Group (JA-SIG) at <http://www.ja-sig.org>.

Teri Olsen is Director of the Web Resource Center for the University of Utah Health Sciences Center. Previously she served as Manager of Web & Communication for Administrative Computing Services, where she managed the migration to Java for student systems.