2003

The Extreme Software Development Series: An Open Curricular Framework for Applied Capstone Courses

Konstantin Läufer
Loyola University Chicago, klaeufer@gmail.com

George K. Thiruvathukal
Loyola University Chicago, gkt@cs.luc.edu

The Extreme Software Development Series: An Open Curricular Framework for Applied Capstone Courses

Konstantin Läufer and George K. Thiruvathukal
Department of Computer Science
Loyola University Chicago
6525 N. Sheridan Road
Chicago, IL 60626, USA
{laufer,gkt}@cs.luc.edu

ABSTRACT

We describe an open, flexible curricular framework for offering a collection of advanced undergraduate and graduate courses in software development. The courses offered within this framework are further unified by combining solid foundations with current technology and play the role of capstone courses in a modern software development track. Our initiative has been very successful with all stakeholders involved.

Category and Subject Descriptors

K.3.2 Computer and Information Science Education; I.7.2 Document Preparation; D.1.3 Concurrent Programming; D.2.11 Software Architectures

General Terms

Design, Languages, Performance

Keywords

Markup languages, XML, Event-based programming, Design patterns, Frameworks, Distributed programming, Protocols, Servlets, Web applications, Enterprise applications

1. INTRODUCTION AND INSTITUTIONAL CONTEXT

We describe a curricular initiative marketed to students under the overarching title *The Extreme Software Development Series*. This initiative was initiated, planned, designed, and implemented by two faculty members over the last three semesters. Our key objective was to provide an open, flexible framework for offering an exciting collection of advanced undergraduate and graduate courses in software development that combine the *state-of-the-art* in research with the *state-of-the-practice* [4] in industry. Both faculty members are active researchers in systems and have industry experience.

Currently, the series consists of the following courses in the authors' areas of expertise; we will describe these courses below in detail.

- Extreme Markup Languages
- Extreme Concurrency
- Extreme Distributed Computing
- Extreme Server-Side Applications

Additional courses in the area of software development will be added to this framework as they become available.

The challenge has been to offer the Extreme Series within our institutional context, a small but research-active department in a liberal-arts-oriented, private, religious university. The department offers an undergraduate major in computer science, a terminal masters degree, as well as a combined five-year bachelors/masters degree. Because of the large set of liberal arts requirements at the undergraduate level, there is not too much room for the computer science major requirements, so flexibility is key. Therefore, the courses offered through this framework are loosely coupled electives that play the role of capstone courses in a modern software development track [1, 3]. Undergraduate students are able to take up to three, and graduate students four, of the courses offered.

Our initiative has been very successful with students, as documented by consistently high enrollments and very high teaching evaluations, possibly the two highest in the department; and with employers, as documented by anecdotal evidence.

2. UNIFYING THEMES OF THE FRAMEWORK

Although the courses offered as part of the Extreme Series cover a wide range of topics, they are designed to form a cohesive series unified by several pervasive themes.

Foundations

All courses are based upon state-of-the-art, theoretical foundations in the relevant areas of research, including:

- programming language theory
- concurrency
- distributed programming

Concepts
All courses study the general, technology-independent concepts relevant to the topics studied, including:

- object-orientation and aspect-orientation
- software architecture, frameworks, and design patterns
- processes and threads
- interaction among distributed objects

Technologies and Tools
All courses are heavily oriented toward “real-world” projects and use state-of-the-practice approaches and technologies as delivery vehicles, including:

- development methodologies, especially agile approaches such as Extreme Programming (XP)
- modeling tools, such as the Unified Modeling Language (UML)
- language technologies, such as Java, Python, and XML
- (mostly) open-source development tools, such as Ant, CVS, Eclipse, JBuilder, and JBoss

Where appropriate, courses include guest lectures by experts from industry.

Place in Curriculum
All courses are consistently positioned in the curriculum as applied electives with common characteristics:

- providing a combination of challenge and excitement
- offered at least once a year (evenly distributed between fall and spring semesters)

From the point of view of other interested faculty, the framework is *open* to the addition of suitable courses that can be presented consistently with the existing ones.

3. COURSE DESCRIPTIONS
In this section, we provide details on each course in the following format: overview, prerequisites, textbooks and other materials, example, and road map (concepts, research/technologies, and projects).

3.1 Extreme Markup Languages
This course covers markup languages and their applications. Specifically, this course discusses XML, XSLT, and the various W3C specifications for manipulating XML documents programmatically, including the DOM and SAX frameworks. The course also covers some advanced topics, including how to manage large XML documents and integration with databases. The course includes several intermediate to advanced programming projects using Python and available XML frameworks. The road map for this course is shown in Table 1.

Prerequisites
The required prerequisite for this course is our version of CS2 and basic or working knowledge of object-oriented programming.

Textbooks
For learning Python, [9] contains two chapters of comprehensive nature to learn the language quickly. A comprehensive and concise tour of XML is presented in [5]. Both texts are recommended but not required. Several excellent tutorials on Python are provided at the Python web site (http://www.python.org/).

Example
In the most recent offering, a distributed calendaring system is being developed using XML. Such a system allows virtually every aspect of XML and its component frameworks to be explored. The end product is one that can be used for real-world project management.

3.2 Extreme Concurrency
This course provides an architectural perspective on the development of concurrent software with an emphasis on design patterns, composition, and reuse. Initially, this course was developed jointly by the first author and another colleague as an avenue to transfer recent research results in concurrency into the classroom, but from a decidedly applied perspective. The course was first offered in 1997 and has evolved significantly since that time, along with the body of research in this area and the resulting technologies. The road map for this course is shown in Table 2.

Prerequisites
The required prerequisite for this course is our version of CS2 and our intermediate object-oriented programming course that follows CS2.

Textbooks
This course uses two texts, one on Java Swing [7] and one on concurrent programming in Java from a design patterns perspective [6]. In addition, the course relies on various online resources. The Swing text is optional because there is high-quality online material available on this subject.

Example
A distributed voting system with the following functional requirements:

- Multiple polling booths. Each polling booth has a button for each candidate.
- Multiple tracking stations. Each tracking station has a vote count display for each candidate.
- There is a single control panel, which is used to open or close the poll, add or remove candidates, and control the frequency of vote count updates.

The voting system has the following nonfunctional requirements:

- Each component follows a presentation-translation-application architecture.
- Remote method invocation (RMI) is used for communication among components.
- There is a central server object providing a passive application data model and a communication router.

3.3 Extreme Distributed Computing

Extreme distributed computing is positioned as a modern discussion of distributed computing systems. In this course, we explore the advanced distributed computing technologies and the important principles and patterns behind them (the tradition of distributed systems): the various forms of transparency, remote procedure calls (RPCs), consistency, coherence, transactions, and security. The actual technologies to be explored will vary but will include a discussion of many Java technologies, including Servlets, Remote Method Invocation (RMI) and Activation, Jini, JNDI, messaging systems, etc. We will also consider the potential for parallel and cluster computing to address server performance. The road map for this course is shown in Table 3.

Prerequisites

The required prerequisite for this course is our version of CS2.

Textbooks

The required textbook is [2], which is used as a reference to supplement the course lectures. Many of the lectures are based on material available on the internet, such as the DNS RFC from http://www.ietf.org/, which is one of the most well-known communities for distributed software applications.

Projects

This course is project-based. Because distributed systems involves more technologies and techniques than could ever be covered in a single course, students are required to investigate technologies presented at a high-level and propose a distributed project that meets the requirements of a distributed system (e.g. support for basic transparency principles, such as fault tolerance, consistency, etc.)

Example

Many projects have been done in the past, including a framework for peer-to-peer computing on clusters of workstations, a fault-tolerant chat system, web caching, and many others.

3.4 Extreme Server-Side Applications

This course provides an architectural perspective on the development of interactive server-based software, building on top of the perspective established in the *Extreme Concurrency* course. This course was first offered in 2000, and recently developed server-side technologies have continually been added to the course. The road map for this course is shown in Table 4.

Prerequisites

The required prerequisite for this course is at least one of *Extreme Concurrency* and *Operating Systems*. In addition, a recommended prerequisite is *Extreme Markup Languages*.

Textbooks

This course requires one text on Java 2 Enterprise architecture [8], which provides a high-level, architectural view of the material. In addition, the course relies on various online resources for design patterns and specific technologies.

Project

A server-based, device-independent bug or issue tracking system, usable with ordinary desktop web browsers, as well as small-screen devices, such as mobile phones or wireless PDAs. This project proceeds in three consecutive phases:

- Static version: a static web site that illustrates all possible scenarios (use cases) of the system. This version uses XHTML and CSS.
- Non-persistent dynamic version: a dynamic web application that is functional but does not include persistent data. The web pages from the previous version become the view templates for this version. This version uses JSPs in conjunction with the Struts web application framework and the Java Mail API for notification.
- Persistent dynamic version: an enterprise application that is functional and includes persistent data. The previous version becomes the web tier for this version. This version uses entity EJBs for persistent storage.

4. DISSEMINATION AND REPLICATION

The authors are committed to disseminating their materials through the following channels:

- All course materials developed by the authors for their students are available on the web. URLs to the course pages are available upon request.
- Course pages include links to the software used. All software supports the most common platforms (Linux, Mac OS X, Windows) and is either open-source or otherwise freely available.
- Sample exams/quizzes and solutions are available to qualified instructors upon request.

Furthermore, the authors are interested in establishing a dialog with colleagues at other institutions who are planning to develop similar courses.

Concept	Research/Technology	Project
Ad-Hoc Document Parsing	Brute-Force Parser	Calendar, To-Do List
Well-formedness and Validation	Basic XML Parsers, DTD	Calendar, To-Do List in XML w/DTD
Document Trees	DOM, Instructor's Framework	Calendar, To-Do List with Recurring Dates, Excluding Dates
Parsing Events	SAX	XElement: An OO Tree Framework Grounded in Design Patterns
Directory Metaphor	XPath	N/A, Part of Later Project
Multiple XML Namespaces	Namespaces, XML/Schema	N/A, Part of Later Project
Transformation	XSLT, Instructor's Framework	Generating XHTML Calendar View
Remote Procedure Call	XML/RPC, SOAP	Distributed Data Collector for a Computing Cluster
XML Frameworks	RSS, Selected Others	Syndication of News, Blogs, CMS Content

Table 1: Road map for Extreme Markup Languages

Concept	Research/Technology	Project
Behavioral modeling	UML state diagrams	Wrist watch model
Graphical user interfaces	Java Swing, JBuilder	Wrist watch implementation
Event-based programming	Java Beans	Voting system
Multi-threading	Java threads and monitors	Adventure game
Reusable components	util.concurrent library	Adventure game
Abstract combinators for concurrency	Instructor's own framework	Microwave oven
Event-based testing	Temporal logic	Microwave oven
Remote communication	Java RMI	Voting system

Table 2: Road map for Extreme Concurrency

Concept	Research/Technology
Transparency	Principles only; numerous examples such as HTTP, DNS, RPC, etc. are introduced to see the big picture.
Networking	TCP/IP, use of UDP for heartbeats and one-way messaging
Coordination	Instructor's Memo System, JavaSpaces
Directories	LDAP (Lightweight Directory Access Protocol)
Remote Procedure Call	RPC Classic (ONC/RPC), Remote Method Invocation (RMI), XML/RPC, SOAP
Fault Tolerance	Database Replication/Mirroring in MySQL or Oracle
Transactional Semantics	Servlet Session Management, Database Transaction Facilities
Parallelism	Parallel Database Servers, Clustering, Round-Robin DNS
Security and Authentication	SSL, Certificate Management, Survey of Authentication Frameworks (Kerberos, Windows Active Directory)

Table 3: Road map for Extreme Distributed Computing

Concept	Research/Technology	Project
Server-based multi-tiered systems	J2EE	Semester-long project: server-based issue-tracking system
Presenting content	XML-based markup languages	Static prototype of project
Modeling dynamic behavior	UML state diagrams	UML state diagram to accompany static prototype
Dynamic web-based applications	Java servlets, JSP	Web tier of project (without persistence)
Web application frameworks	Struts	Web tier of project (without persistence)
Server-side business logic components	Session EJBs	Not required in project
Server-side persistent components	Entity EJBs, Hibernate	Complete multi-tier project (with persistence)
Security and authentication	J2EE web container authentication	Complete project
Application servers	JBoss	Deployment of project

Table 4: Road map for Extreme Server-Side Applications

5. CONCLUSION AND NEXT STEPS

In conclusion, the curricular framework presented here has turned out to be a success for all stakeholders: administrators, faculty, students, and employers.

The challenge for the coming years is to keep the composition of the curricular framework up-to-date with new research and technologies in software development. This is greatly facilitated when participating faculty are active researchers and have ongoing relationships with industry.

Within the bigger context of our academic department, and encouraged by other authors [3], our plan is to evolve the Extreme Software Development Series into the centerpiece of a full-fledged undergraduate major and graduate specialization in software development.

6. REFERENCES

[1] Eric Allen, Robert Cartwright, and Charles Reis. Production programming in the classroom. In *Proceedings of the 34th SIGCSE technical symposium on Computer science education*, pages 89–93. ACM Press, 2003.

[2] George Coulouris, Jean Dollimore, and Tim Kindberg. *Distributed Systems: Concepts and Design*. Addison-Wesley, 3rd edition, 2000.

[3] Alan Fekete and Bob Kummerfeld. Design of a major in software development. In *Proceedings of the 33rd SIGCSE technical symposium on Computer science education*, pages 73–77. ACM Press, 2002.

[4] Robert L. Glass. Practical programmer: On personal technical obsolescence. *Communications of the ACM*, 43(7):15–17, 2000.

[5] Cheryl M. Hughes. *The Web Wizard's Guide to XML*. Addison Wesley, 2002.

[6] Doug Lea. *Concurrent Programming in Java: Design Principles and Patterns*. The Java Series. Addison-Wesley, 2nd edition, 2000.

[7] David Karr Matthew Robinson, Pavel Vorobiev. *Swing*. Manning, 2nd edition, 2003.

[8] Inderjeet Singh, Beth Stearns, and Mark Johnson. *Designing Enterprise Applications with the J2EE Platform*. The Java Series. Addison-Wesley, 2nd edition, 2002.

[9] George K. Thiruvathukal, John P. Shafaee, and Thomas W. Christopher. *Web Programming in Python*. Prentice Hall PTR, 2002.