CS 5984: Object-Oriented Systems and Languages
Fall 2008 Syllabus

Meeting Times: Monday, 3:30-4:45pm

Meeting Place: TBA

Instructor: Dr. Eli Tilevich, 213 Knowledge Works II, 540-231-3475
tilevich@cs.vt.edu
Office Hours: TBA

Description: Object-oriented (OO) systems has been one of the most dynamic research areas in recent years. Beyond encapsulation, inheritance, and polymorphism, research in OO systems is a confluence of various topics in systems, programming languages, compilers, and software engineering. Specifically, research in OO systems has engendered several exciting recent developments in areas including: programming languages (e.g., genericity, reflection, meta-programming, bytecode engineering, virtual dispatch, garbage collection, just-in-time-compilation), middleware (e.g., distributed-object systems), concurrency (e.g., Java memory model), and many others. Several novel programming paradigms such as aspect-oriented programming (AOP) stem from research in object-oriented technolgies. Therefore, knowledge in OO systems is essential for anyone involved in development of next generation technologies.

This course will provide students with a background in OO technologies by covering both standard research literature and providing hands-on experience with specific technologies. In addition, the course will introduce students to research opportunities in current state-of-the-art OO systems. Additional topics covered will be determined by the individual interests of the class's participants.

Prerequisites: Because this is a graduate course, prerequisites are not strictly enforced. However, you should have knowledge of programming languages equivalent to an undergraduate PL survey course and fluency in at least one OO language.

Evaluation:

hands-on exercises	(20%)
a midterm exam	(20%)
a research paper presentation	(10%)
a term paper or project	(40%)
class participation	(10%)

Other Resources:

Listserv: TBA
Web Page:TBA

Course Outline:

The course will cover an extensive sample of work from the Object-Oriented Systems and Languages literature such as can be found in the proceeding of the OOPSLA and ECOOP conferences. Specific areas to be covered include:

Design Patterns—a critical view and analysis (not tutorial)

Component-based designs: layered design, mixin, mixin layers

Aspect-Oriented Programming, Subject-Oriented Programming, Adaptive Programming

OO Type Systems: parameterization mechanisms for Java, virtual types, module systems

Language Extensibility: meta-object protocols, reflection

Implementation issues: efficient dynamic dispatch, multiple inheritance and object layout, garbage collection for Java

OO Distributed Systems: OO middleware, distributed object systems

Software tools: code generation, code transformation, bytecode engineering

Reading Material:

There is no textbook for this course. Papers from the literature will be used. The reading list, below, offers a sampling of selected papers and books. The list is not complete (does not include very recent papers) but it is meant to give you a taste of the material we will study.

Reading List:

O. Agesen, S. Freund, and J. Mitchell, "Adding Type Parameterization to the Java Language", *OOPSLA 1997*, 49-65.

Bowen Alpern, Anthony Cocchi, Stephen Fink, David Grove, and Derek Lieber, "Efficient Implementation of Java Interfaces: Invokeinterface Considered Harmless," in *OOPSLA 2001*.

Henri E. Bal and M. Frans Kaashoek, "Object Distribution in Orca Using Compile-Time and Run-Time Techniques," *OOPSLA 1993*.

D. Batory, V. Singhal, M. Sirkin, and J. Thomas, "Scalable Software Libraries", *ACM SIGSOFT* 1993.

John K. Bennett, "The Design and Implementation of Distributed Smalltalk," in *OOPSLA 1987*.

T.J. Biggerstaff, "The Library Scaling Problem and the Limits of Concrete Component Reuse", *3rd Int. Conf. on Softw. Reuse (ICSR '94)*.

G. Bracha and W. Cook, "Mixin-Based Inheritance", *ECOOP/OOPSLA 1990*, 303-311.

G. Bracha, M. Odersky, D. Stoutamire and P. Wadler, "Making the future safe for the past: Adding Genericity to the Java Programming Language", *OOPSLA 1998*.

K.B. Bruce, M. Odersky, and P. Wadler, "A Statically Safe Alternative to Virtual Types", *ECOOP 1998*.

L. Cardelli and P. Wegner, On Understanding Types, Data Abstraction, and Polymorphism, *Computing Surveys*, 17(4): Dec 1985, 471-522.

S. Chiba, “Open C++ Programmer's Guide for Version 2”, SPL-96-024, Xerox PARC, 1996.

K. Czarnecki and U. Eisenecker. *Generative Programming: Methods, Techniques, and Applications*. Addison-Wesley, 2000.

K. Czarnecki and U. Eisenecker, "Synthesizing Objects", *ECOOP 1999*, 18-42.

M.A. Ellis and B. Stroustrup, *The Annotated C++ Reference Manual*, Addison-Wesley, 1990.

Michael Factor, Assaf Schuster and Konstantin Shagin, “Instrumentation of Standard Libraries in Object-Oriented Languages: the Twin Class Hierarchy Approach,” in *OOPSLA 2004*.

R.B. Findler and M. Flatt, “Modular Object-Oriented Programming with Units and Mixins”, *Int. Conf. on Functional Programming*, 1998.

M. Flatt, S. Krishnamurthi, M. Felleisen, “Classes and Mixins”. ACM *Symposium on Principles of Programming Languages*, 1998 (PoPL 98).

I.R. Forman, S. Danforth, and H. Madduri, “Composition of Before/After Metaclasses in SOM”, *OOPSLA 1994*.

E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.

J. Goguen, “Reusing and interconnecting software components”, *IEEE Computer*, February 1986, 16-28.

James Gosling, Bill Joy, Guy L. Steele, *The Java Language Specification*, Addison-Wesley, Reading, Massachusetts, 1996.

W. Harrison and H. Ossher, “Subject-Oriented Programming (A Critique of Pure Objects)”. *OOPSLA 1993*, 411-428.

R. Helm, I. Holland, and D. Gangopadhyay, “Contracts: Specifying Behavioral Compositions in Object-Oriented Systems”. *OOPSLA 1990*, 169-180.

I. Holland, “Specifying Reusable Components Using Contracts”, *ECOOP 1992*, 287-308.

R. Johnson and B. Foote, “Designing Reusable Classes”, *Journal of Object-Oriented Programming*, 1(2): June/July 1988, 22-35.

R. Keller, U. Hoelzle, “Binary Component Adaptation”, *ECOOP 1998*.

G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J. Loingtier, and J. Irwin, “Aspect-Oriented Programming”, *ECOOP 1997*, 220-242.

G. Kiczales, J. des Rivieres, and D. G. Bobrow, *The Art of the Metaobject Protocol*, MIT Press, 1991.

Joerg Kienzle and Rachid Guerraoui, “AOP: Does It Make Sense? The Case of Concurrency and Failures,” *ECOOP 2002*.

K.J. Lieberherr, *Adaptive Object-Oriented Software: The Demeter Method with Propagation Patterns*, PWS Publishing Company, Boston, 1996.

O. L. Madsen and B. Møller-Pedersen, “Virtual classes: A powerful mechanism in object-oriented programming”, *OOPSLA 1989*, 397-406.

O. L. Madsen, B. Møller-Pedersen, and K. Nygaard, *Object-Oriented Programming in the BETA Programming Language*. Addison-Wesley, 1993.

M. Mezini, “Dynamic Object Evolution without Name Collisions”, *ECOOP 97*, 190-219.

M. Mezini and K. Lieberherr, “Adaptive Plug-and-Play Components for Evolutionary Software Development”, *OOPSLA 1998*.

D.A. Moon, “Object-Oriented Programming with Flavors”, *OOPSLA 1986*.

A. Myers, J. Bank and B. Liskov, “Parameterized Types for Java”, ACM *Symposium on Principles of Programming Languages*, 1997 (PoPL 97).

M. Odersky and P. Wadler, “Pizza into Java: Translating theory into practice”, *ACM Symposium on Principles of Programming Languages*, 1997 (PoPL 97).

H. Ossher and W. Harrison, “Combination of Inheritance Hierarchies”, *OOPSLA 1992,* 25-40.

H. Ossher, M. Kaplan, W. Harrison, A. Katz, and V. Kruskal, “Subject-Oriented Composition Rules”, *OOPSLA 1995,* 235-250.

C. Prehofer, “Feature-Oriented Programming: A Fresh Look at Objects”, *ECOOP 1997*, 419-443.

Francisco Reverbel and Marc Fleury, “The JBoss Extensible Server,” in *ACM Middleware 2003 Conference*.

Y. Smaragdakis and D. Batory, “Implementing Reusable Object-Oriented Components”, *5th Int. Conf. on Softw. Reuse (ICSR ‘98)*, IEEE Computer Society Press, 1998.

Y. Smaragdakis and D. Batory, “Implementing Layered Designs with Mixin Layers”, *ECOOP 1998*.

L. Seiter, J. Palsberg, and K. Lieberherr, “Evolution of Object Behavior using Context Relations”, *ACM SIGSOFT* 1996.

Silicon Graphics Computer Systems Inc., *STL Programmer’s Guide*. See: http://www.sgi.com/Technology/STL/ .

C. Simonyi, “The Death of Computer Languages, the Birth of Intentional Programming”, *NATO Science Committee Conference,* 1995.

Sergio Soares, Eduardo Laureano, Paulo Borba, “Implementing Distribution and Persistence Aspects with AspectJ,” *OOPSLA 2002*.

A. Stepanov and M. Lee, “The Standard Template Library”, ANSI/ISO C++ Standard.

P. Steyaert, W. Codenie, T. D'Hondt, K. De Hondt, C. Lucas, and M. Van Limberghen, “Nested Mixin-Methods in Agora”, *ECOOP 1993*, 197-219.

B. Stroustrup, *The C++ Programming Language, 3rd Ed.*, Addison-Wesley, 1997.

Michiaki Tatsubori, Toshiyuki Sasaki, Shigeru Chiba, and Kozo Itano, “A Bytecode Translator for Distributed Execution of ‘Legacy’ Java Software,” *ECOOP 2001*.

K. Thorup, “Genericity in Java with Virtual Types”, *ECOOP 1997*, 444-471.

Eli Tilevich and Yannis Smaragdakis, “J-Orchestra: Automatic Java Application Partitioning,” *ECOOP 2002*.

Eli Tilevich, Stephan Urbanski, Yannis Smaragdakis, and Marc Fleury, “Aspectizing Server-Side Distribution,” *in Automated Software Engineering (ASE 2003)*.

Eli Tilevich and Yannis Smaragdakis, "Portable and Efficient Distributed Threads for Java," in *ACM/IFIP/USENIX Middleware 2004*.

Eli Tilevich and Yannis Smaragdakis, “NRMI: Natural and Efficient Middleware,” in *ICDCS 2003.*

M. VanHilst and D. Notkin, “Using Role Components to Implement Collaboration-Based Designs”, *OOPSLA 1996.*

Jim Waldo, Geoff Wyant, Ann Wollrath, and Sam Kendall, “A note on distributed computing,” Technical Report, Sun Microsystems Laboratories, SMLI TR-94-29, Nov. 1994.

Ann Wollrath, Roger Riggs, and Jim Waldo, “A Distributed Object Model for the Java System,” in *USENIX 1996 Conference on Object-Oriented Technologies, pages 219-232, Toronto, Ontario, Canada, June 1996.*

David Zook, Shan Shan Huang, and Yannis Smaragdakis, “Generating AspectJ Programs with Meta-AspectJ,” in *Generative Programming and Component Engineering Conference (GPCE), 2004* .