# MATHIAS GUENTER RICKEN

6100 Main Street MS-132 mgricken@rice.edu
Houston, TX 77005-1892 (713) 835-2446

**NOTE** **I am currently not looking for employment.**

**EDUCATION**

**Ph.D. in Computer Science** expected 2011
Rice University, Houston, TX. Graduate GPA 3.98/4.00
Research Area: Programming Languages. Advisor: Dr. Robert Cartwright

**M.S. in Computer Science** October 2007
Rice University, Houston, TX
Thesis: "A Framework for Testing Concurrent Programs"
Republished 2009 by VDM Verlag (ISBN 978-3-639-15074-2)

**B.S. in Computer Science** May 2004, magna cum laude
Rice University, Houston, TX. GPA 3.89/4.00

**Abitur** 1999. Average 1.0/1.0. Ranked 4th in the state of Bremen
Hermann Boese Gymnasium, Bremen, Germany

Computer skills: C, C++, C#, Java, Promela/SPIN, Assembly, Scheme, OCaml
Languages spoken fluently: English, German

**PUBLICATIONS**

**Mint: Java Multi-stage Programming Using Weak Separability**
Westbrook, E, M. Ricken, J. Inoue, Y. Yao, T. Abdelatif, and W. Taha
*Proceedings of the 2010 ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 2010), ACM 2010*
"Multi-stage programming (MSP) provides a disciplined approach to run-time code generation. In the purely functional setting, it has been shown how MSP can be used to reduce the overhead of abstractions, allowing clean, maintainable code without paying performance penalties. Unfortunately, MSP is difficult to combine with imperative features, which are prevalent in mainstream languages. The central difficulty is scope extrusion, wherein free variables can inadvertently be moved outside the scopes of their binders. This paper proposes a new approach to combining MSP with imperative features that occupies a 'sweet spot' in the design space in terms of how well useful MSP programs can be expressed and how easy it is for programmers to understand. The key insight is that escapes (or 'anti-quotes') must be weakly separable from the rest of the code, i.e. the computational effects occurring inside an escape that are visible outside the escape are guaranteed to not contain code. To demonstrate the feasibility of this approach, we formalize a type system based on Lightweight Java which we prove sound, and we also provide an implementation, called Mint, to validate both the expressivity of the system and the performance gains attainable by using MSP in this setting."

**PUBLICATIONS** (continued)

**Test-First Java Concurrency for the Classroom**
Ricken, M., and R. Cartwright
*Proceedings of the Forty-First SIGCSE Technical Symposium on Computer Science Education. ACM, 2010*
"Concurrent programming is becoming more important due to the availability of multi-core processors and the prevalence of graphical user interfaces (GUIs). To adequately prepare students for the concurrent future, instructors have begun to address concurrency even in introductory courses. Unfortunately, practices like test-driven development that give students a safe footing in single-threaded environments do not extend well into the concurrent domain. This paper describes how ConcJUnit can simplify writing unit tests for multi-threaded programs, and provides examples that can be used to introduce students to concurrent programming."

**ConcJUnit: Unit Testing for Concurrent Programs**
Ricken, M., and R. Cartwright
*Proceedings of the 7th International Conference on the Principles and Practice of Programming in Java (PPPJ 2009)*
*ACM International Conference Proceeding Series, ACM, 2009*
"We present ConcJUnit, an extension of the popular unit testing framework JUnit that simplifies the task of writing tests for concurrent programs by handling uncaught exceptions and failed assertions in all threads, and by detecting child threads that were not forced to terminate before the main thread ends."

**A Framework for Testing Concurrent Programs**
*M.S. Thesis, October 2007*
*Republished 2009 by VDM Verlag* (ISBN 978-3-639-15074-2)
"To facilitate the development of concurrent programs, we are developing: (1) An extension of the JUnit framework that actively supports the developer by treating tests that could silently ignore failures in auxiliary threads as test errors; (2) A lightweight Java annotation language that can be used to specify and check the threading invariants of both existing and new code; (3) A testing framework that can record and analyze the schedules of unit tests, detect deadlocks, and run the tests using modified schedules, increasing the likelihood that concurrency problems are discovered."

**Nifty Assignment: Temperature Calculator – Programming for Change**
Nguyen, D., and M. Ricken
*Proceedings of the Fifteenth OOPSLA Educators' Symposium. ACM, 2006*
"Programming for change is a continual process in which software is designed over many iterations to capture the problem's essence and express. At the heart of this process is the effort to identify those elements that can vary (variants) and delineate them from those that do not – the invariants. A properly designed software system should strive to decouple the variants from the invariants in order to facilitate the re-use of the invariants and allow modifications to the variants with minimal perturbation to the existing code."

**PUBLICATIONS** (continued)

**Design Patterns for Parsing**
Nguyen, D., M. Ricken, and S. Wong
*Proceedings of the Thirty-Sixth SIGCSE Technical Symposium on Computer Science Education. ACM, 2005*
"We provide a systematic transformation of an LL(1) grammar to an object model that consists of (1) an object structure representing the non-terminal symbols and their corresponding grammar production rules; and (2) a union of classes representing the terminal symbols (tokens).
We present a variant form of the visitor pattern and apply it to the above union of token classes to model a predictive recursive descent parser on the given grammar. Parsing a non-terminal is represented by a visitor to the tokens. For non-terminals that have more than one production rule, the corresponding visitors are chained together according to the chain of responsibility pattern in order to be processed correctly by a valid token. The abstract factory pattern, where each concrete factory corresponds to a non-terminal symbol, is used to manufacture appropriate parsing visitors.
Our object-oriented formulation for predictive recursive descent parsing eliminates the traditional construction of the predictive parsing table and yields a parser that is declarative and has minimal conditionals. It not only serves to teach standard techniques in parsing but also as a non-trivial exercise of object modeling for objects-first introductory courses."

**Nifty Assignment: Marine Biology Simulation**
Cheng, E., D. Nguyen, M. Ricken, and S. Wong
*Proceedings of the Thirteenth OOPSLA Educators' Symposium. ACM, 2004*
"The Marine Biology Simulation is designed as a final project in an objects-first CS2 course. It provides an entertaining setting that serves as compelling example of the powers of object-oriented design and programming."

**Nifty Assignment: Abstract Factories and the Shape Calculator**
Cheng, E., D. Nguyen, M. Ricken, and S. Wong
*Proceedings of the Thirteenth OOPSLA Educators' Symposium. ACM, 2004*
"The Shape Calculator is an assignment targeted at CS1 students in an objects-first curriculum. It can serve as a powerful yet entertaining example of the advantages of object-orientation."

**Design Patterns for Marine Biology Simulation**
Nguyen, D., M. Ricken, and S. Wong
*Proceedings of the Thirty-Fifth SIGCSE Technical Symposium on Computer Science Education. ACM, 2004*
"We specify and implement a GUI application that simulates marine biological systems by making extensive use of object-oriented design patterns.
The key design patterns are model-view-control, observer/observable, visitor, command, factory method and decorator. These design patterns help delineate the roles and responsibilities of the objects in the system, establish loose coupling between objects and arrange for the objects to communicate and cooperate with one another at the highest level of abstraction. The result is an application that exhibits minimal control flow, yet is powerful, robust, flexible and easy to maintain.
Our work entails a non-trivial redesign of the current AP Computer Science Marine Biology Simulation case study and may serve as a case study for an introductory 'object-first' curriculum."

**PRESENTATIONS**

**Agile and Efficient Domain-Specific Languages using Multi-stage Programming in Java Mint**
Ricken, M., E. Westbrook, and W. Taha
*Ninth International Conference on Generative Programming and Component Engineering (GPCE'10). ACM, 2010*
"Domain-specific languages (DSLs) are a powerful productivity tool because they allow domain experts, who are not necessarily programming experts, to quickly develop programs. DSL implementations have unique constraints for programming languages because they must be efficient, in order to ensure high productivity, but they must also be agile, in order to meet the rapidly changing demands of their domains. In this tutorial we show how multi-stage programming (MSP) can be used to build staged interpreters, which combine the agility of interpreters with the efficiency of compilers. The tutorial is conducted in Java Mint, an multi-stage Java based on recent work incorporating MSP into imperative object-oriented languages. In the first half of the tutorial, we introduce MSP by demonstrating how to write a staged interpreter for a number of basic language constructs, such as recursive functions, conditionals, and let expressions. In the second half, we extend our staged interpreter to take advantage of several well-known compiler optimizations, including type inference, constant folding, and static parallel loop scheduling. We highlight the opportunities afforded by using MSP with object-oriented design to quickly create efficient DSL implementations."

**Mint: A Multi-stage Extension of Java**
Westbrook, E, M. Ricken, J. Inoue, Y. Yao, T. Abdelatif, and W. Taha
*Purdue University Computer Science Colloquia, March 15, 2010*
(see PLDI 2010 publication for description)

**Object-Oriented Design Festival Workshop**
Cheng, E., D. Nguyen, M. Ricken, and S. Wong
*Thirty-Seventh SIGCSE Technical Symposium on Computer Science Education. ACM, 2006*
"Object-oriented (OO) programming begins with analysis and design that produce a model describing the objects in the problem domain, their relationships, creation and interactions. The workshop covers fundamentals of OO analysis and design such as abstraction, separation of variants from invariants and decoupling of system components, via appropriate applications of composition, inheritance, polymorphism, and design patterns. The workshop will progress from a small design example illustrating the principles to a larger design problem to be solved by small teams of participants. Their solutions will be discussed in terms of design goals and compared against a solution provided by the presenters."

**TEACHING**

**Customer**, Software Engineering Methodology, Fall 2010
Rice University, Houston, TX
Will act as demanding customer for students in a class that models a realistic software development scenario and that teaches principles of software engineering.

**TEACHING** (continued)

**Mentor**, Independent Study, Fall 2009, Spring 2010
Rice University, Houston, TX
Provided advice and supervision to undergraduate computer science students for independent studies concerned with (1) extending the DrJava development environment, and (2) multi-stage programming.

**Instructor**, Production Programming, Spring 2009
**Teaching Assistant**, Production Programming, 2 semesters
Rice University, Houston, TX
Held all class lectures, designed the curriculum, chose projects for student groups, and assigned final grades. As teaching assistant, maintained website and solutions, helped students with Ant and Subversion, administered SourceForge accounts.

**Instructor**, Principles of Object-Oriented Programming II, Fall 2008
Rice University, Houston, TX
Held all class lectures and laboratory tutorials, modified and designed the curriculum, supervised teaching assistants, graded exams and homework assignments, and assigned final grades.

**Teaching Assistant**, Programming Languages, 3 semesters
Rice University, Houston, TX
Held several class lectures, consulted undergraduate and graduate students, and graded their exams and homework assignments. Assisted in conversion of lectures and assignments to OCaml. Maintained website and solutions, improved grading scripts.

**Teaching Assistant**, Intermediate Programming, 9 semesters
Rice University, Houston, TX
Held several class lectures. Presented weekly tutorials on Unix, Java, design patterns, and tools; consulted college students and graded their exams and homework assignments. Maintained website and grade database.

**EXPERIENCE**

**Developer**, JavaPLT, January 2006 –
Rice University, Houston, TX
Extended and maintained DrJava, an open-source cross-platform Java development environment (~350KLOC), and made it suitable for use on large software projects. Refactored the compiler interface, integrated the NextGen, Habanero Java, and Java Mint research compilers, and developed DrJava into a compiler research platform.

**Research Assistant**, Programming Languages Team, May 2004 –
Rice University, Houston, TX
Investigated and implemented testing tools for concurrent Java programs.
Implemented a multi-stage programming extension of Java called Mint.
Designed and developed course material for computer science courses in object-oriented programming and concurrent programming.

**EXPERIENCE** (continued)

**R & D Intern**, Real-Time and Embedded Systems, May 2003 – August 2003
National Instruments, Austin, TX
Modified the LabVIEW Embedded environment to generate multi-threaded C source code for different operating systems and hardware platforms.

**Software Developer**, Programming Languages Team, August 2002 – May 2003
Rice University, Houston, TX
Developed the programming environment DrC#.

**Research Assistant**, Computer Graphics, May 2002 – December 2002
Rice University, Houston, TX
Independently researched and implemented texture and geometry synthesis algorithms in computer graphics; developed applications for a haptic input device.

**HONORS**

Doctoral Fellowship (2004 – 2010)
Rice University/Texas Medical Center Graduate Teaching Certificate
Dean's Teaching Assistant (2008 – 2009)
Sid Richardson Fellow
Rice Undergraduate Scholar
Tau Beta Pi Engineering Honor Society (Officer 2003 – 2004)
Louis J. Walsh Merit Scholarship in Engineering 2001 – 2004
Rice Ambassador, Corps of Special Aides to the Governor of Texas (2001 – 2004)
Rice University President's Honor Roll Fall 2000, 2002, 2003; Spring 2002, 2003

**MEMBERSHIP**

Association for Computing Machinery (ACM) Student Member
Special Interest Group on Programming Languages (SIGPLAN) Student Member
Special Interest Group on Computer Science Education (SIGCSE) Student Member

**ACTIVITIES**

Brian O'Neill's Running Club: Fitness, Fun, Philanthropy
Houston Grand Opera Opening Nights for Young Professionals
Houston Symphony Young Professionals Backstage
Rice Wine Society (Secretary/Treasurer, 2004 – 2006)
Rice Computer Science Club (Vice President, 2003 – 2004)
Rice Engineering Society Council (Secretary/Treasurer, 2003 – 2004)