

Compilation 2011

What Will You Learn?

Jan Midtgaard
Michael I. Schwartzbach
Aarhus University

What You Will Have The Chance to Learn

- **Concrete skills:**
 - compiler technology
 - insights into programming language design
- **Technical skills:**
 - a new programming language: OCaml
 - all about Java
 - building and testing software on a larger scale
 - reading huge and complex specifications
- **Organizational skills:**
 - handling stress and deadlines
 - finding relevant help and information
 - surviving as a group

Compiler Technology

- Compiler architecture
- Typical compiler phases
- Scanners and parsers
- Scope resolvers
- Type checkers
- Static analyzers
- Code templates

- **You can build your own compiler**

Programming Language Design

- The interplay between languages and compilers
- Features that are difficult to implement
- Limitations of current technology
- Scope rules
- Type rules
- Consequences of undecidability

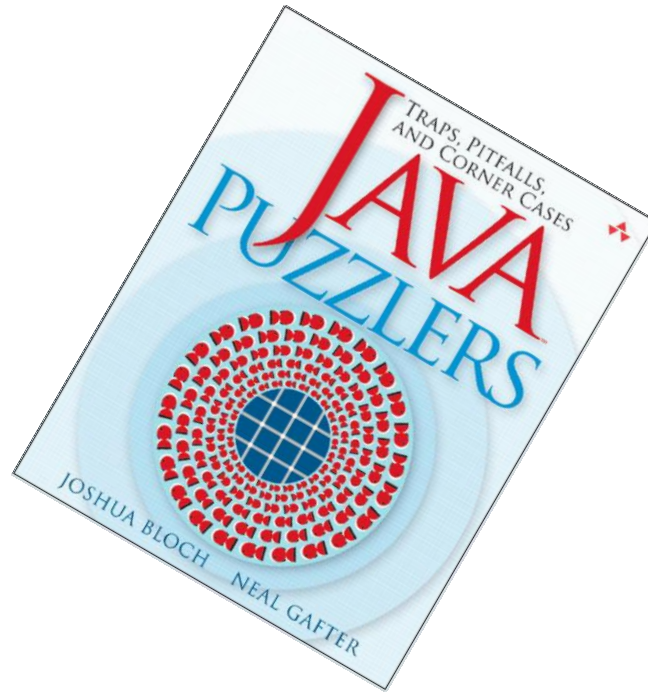
- **You can discuss language design intelligently**
- **You can design another language**

A new programming language: OCaml

- OCaml is the right tool for the job:
it is made for tree processing
 - OCaml is a functional language (like Scheme)
 - OCaml is strongly typed (more than Java)
 - Algebraic datatypes
 - Pattern matching
 - Discourages assignment(!)
-
- **One more language tool in your CS belt**

All About Java

- The subtleties of the Java specification
 - Name resolution
 - Hierarchy checks
 - Type checking
 - Static analysis
 - JVM and verification
 - Code templates
-
- **You are a Java expert, not merely a user**



Building and Testing Software

- 6,000-10,000 lines of code to hand in
 - Code sharing and versioning (`svn`)
 - Using `ocaml`doc
 - Unit testing
 - Multiple recursive traversals
 - Auto-generated code
 - Debugging
-
- **You can handle a complex piece of software**

Reading Complex Specifications

- The Java Language Specification (25,000 lines)
- The JVM Specification (24,000 lines)
- The ocamllex and ocamlyacc specifications

- The Joos Languages
- 6 complex and subtle hand-in specifications

- **You can read huge and complex specs**

Handling Stress and Deadlines

- You have met 7 hard deadlines
- Deadlines are a way of life, not a sudden crisis
- Balancing ambitions with resources
- Realistic estimations of work load
- Getting started in time

- **You are better at facing stress and deadlines**

Finding Help and Information

- Specifications may be missing or unclear
- Don't panic!
- Help is available from many sources
- RTFM
- Help each other
- Use weekly consulting productively
- Ask useful questions on the phorum

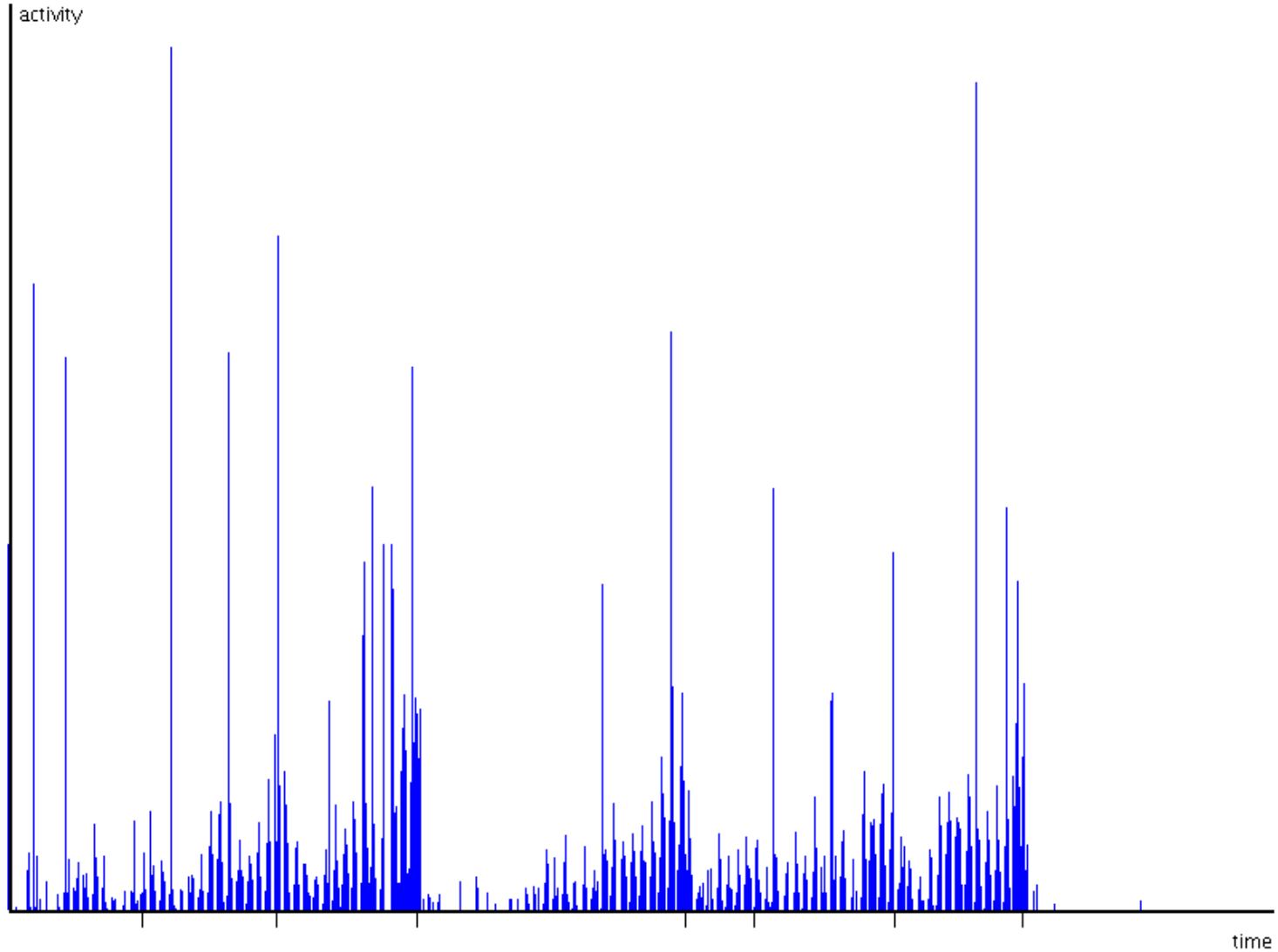
- **You are better at finding help and information**

Surviving as a Group

- Group work is a challenge
- Negotiating ambitions
- Exploiting different skills
- Setting aside your ego
- Respecting internal deadlines
- Ensuring your own outcome

- **You have more experience in group work**

Activity Curve 2009



Surviving as a dOvs student

- Start early
- Attend the lectures
- Ask questions
- Make sure ***you*** understood the material.
- Claim ***your*** share of the lecturers/TA's time.

Background literature recommendations

- Basics of Compiler Design by Torben Æ. Mogensen, available at:
<http://www.diku.dk/hjemmesider/ansatte/torbenm/Basics/>
- Compilers: Principles, Techniques, and Tools (2nd Edition) by Alfred V. Aho, Monica S. Lam, Ravi Sethi, and Jeffrey D. Ullman
- Modern Compiler Implementation in ML/Java by Andrew W. Appel [and Jens Palsberg]
- Advanced Compiler Design and Implementation by Steven S. Muchnick (advanced material)
- Introduction to Objective Caml by Jason Hickey, available at: <http://files.metaprl.org/doc/ocaml-book.pdf>