# The Myth and Reality of using Haskell in the "Real World"

Experiences from darcs

David Roundy

Cornell University

September 24, 2005

## Outline

## SCM: "Source Code Manager"?

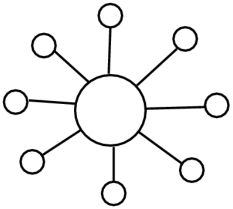
- Numerous acronyms: RCS, SCM, VCS
- Keeps track of changes to source code so you can track down bugs and work collaboratively.
- Most famous example: CVS

## Ideas behind darcs

- A simple "egalitarian" distributed model
- "Cherry picking" of changes
- Avoidance of "merge points"—no history

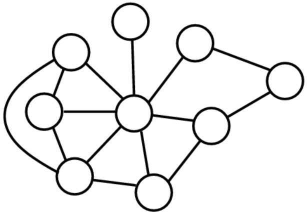
## Distributed rather than centralized

Centralized



Examples: CVS, Subversion, Perforce

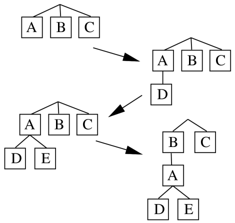
Distributed



Examples: darcs, Git, Bitkeeper, monotone, arch

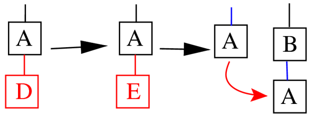
## Change-based rather than version-based

Version-based



Examples: Git, Bitkeeper, Monotone, CVS, Subversion

Change-based



Examples: darcs

## Early darcs history

| | |
|---|---|
| Summer 2002 | First version of darcs written in C++ |
| October 21, 2002 | First commit of darcs written in Haskell |
| April 3, 2003 | Darcs 0.9.3, its first public release |
| July 7, 2003 | Windows support added |

## Haskell vs other languages

Two developers have tried to reimplement darcs in another language:

- yarcs: in python
- jarcs: in java

Neither project ever got to the functionality of the initial release of darcs.

### Advantages of Haskell

- Statically typesafe
- Higher-order functions
- Lazy evaluation
- Monadic syntactic sugar

## Haskell will lead to few contributors

*[...] However, since few developers truly grok functional programming, darcs is less likely to get other developers to help extend it. It does get contributions—a few minor contributions by others have been reported to me—but they're nothing compared to the scale of work by others in Subversion or GNU Arch.*

—David A. Wheeler, May 2005

## Haskell will lead to few contributors

*3. DARCS is written in Haskell. This is not a problem either, but I'd think there are fewer people who can hack Haskell than people who can hack C, C++, Java, Python or similar.*

—Matthias Andree, April 2005

*[...] The only real problem I see with Darcs is that it is written in Haskell, thus reducing the scope of potential contributors / bug squashers to the project ...*

—Julien Ponge, April 2005

## Haskell will lead to few contributors

*The Haskell language is a rather obscure functional language, known to but few, and additionally is reported to characteristically suffer unpredictable but sometimes severe performance problems. Because of the language's obscurity, there have been relatively few third-party contributions to the codebase.*

—Rick Moen

*Drawbacks: How many people can hack Haskell?*

—Rick Moen, September 2005 (LWN)

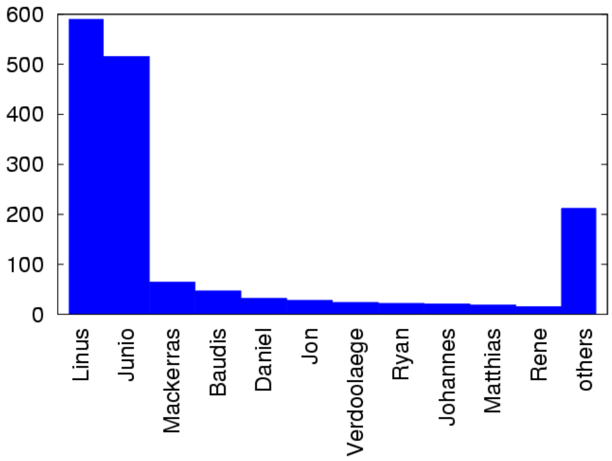
## Darcs has many contributors

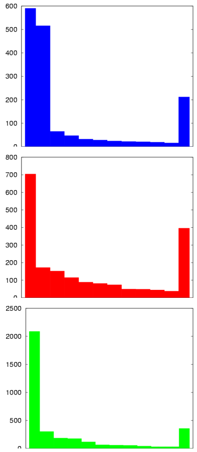
- Darcs has a total of 91 contributors (git has 79).
- Darcs has had 59 contributors this year.
- Many, but not most, are documentation or test suite contributions (i.e. not Haskell).

### Who contributes to darcs? (in Haskell)

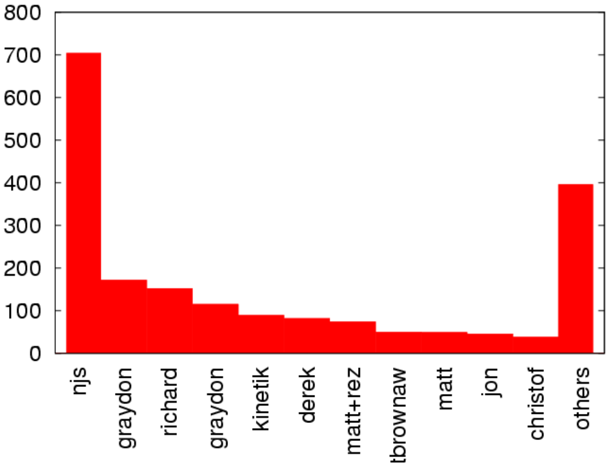
- Haskell coders looking for a project.
- Coders interested in darcs who learn Haskell to contribute.

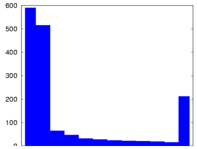
## Contributors to git

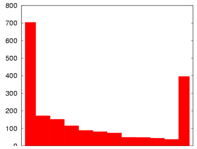


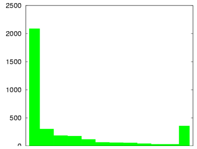


## Contributors to monotone

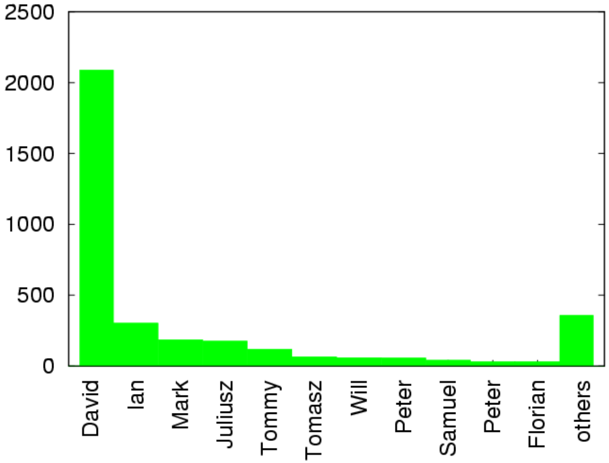


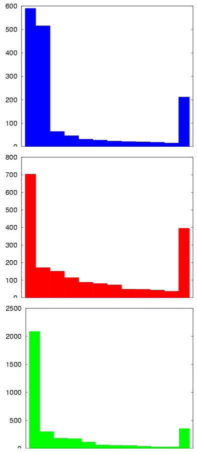




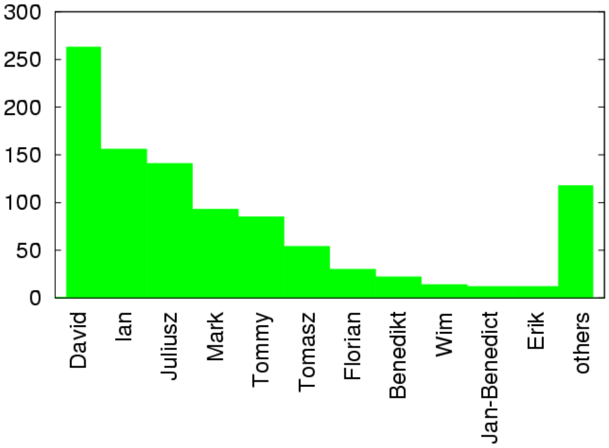


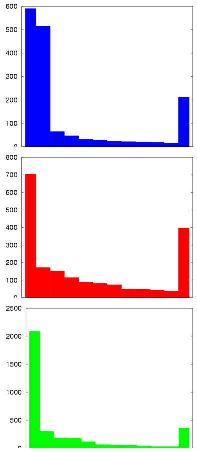
## Contributions over darcs' entire history



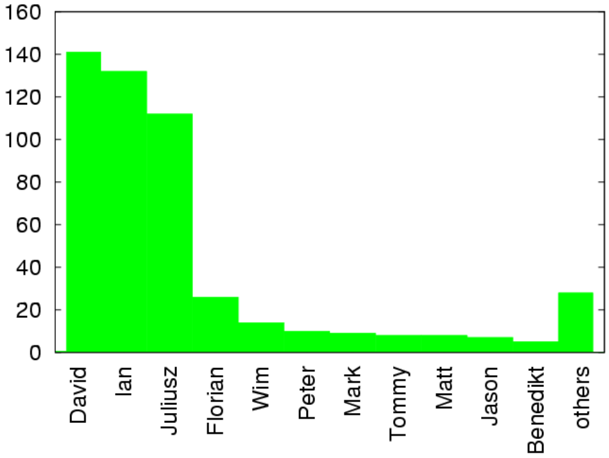


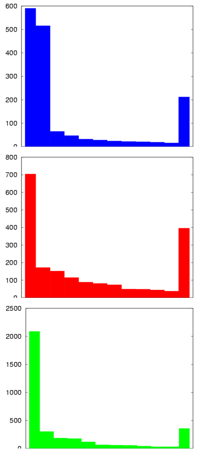
## Contributions to darcs since January 2005





## Contributions to darcs since April 2005





## Haskell code is slow and bloated

*However, while Haskell is intriguing, in my experience programs written in it are generally slow, and possibly worse, its performance is unpredictable.*

—David A. Wheeler, May 2005

*darcs scares me a bit because it's in haskell, I don't believe very much in functional languages for compute intensive stuff, ram utilization skyrockets sometime (I wouldn't like to need >1G of ram to manage the tree).*

—Andrea Arcangeli, February 2005

## Optimization in Haskell

Optimization is definitely trickier than in an ordinary imperative language.

- Laziness makes possible non-invasive space and time optimizations. (e.g. lazy reading, parsing and consumption of patches).
- Laziness makes it very easy to introduce memory and resource leaks.
- Can use the Foreign Function Interface to call C or library functions for low-level operations (e.g. string compare).

## GHC is hard to install

*One possible drawback to it is that it is written in Haskell and has to be compiled with GHC. GHC is a Haskell compiler which is itself written in Haskell, with a problematic bootstrapping procedure, and Haskell may not be entirely scalable due to the limitations of the language.*

—"Better SCM Initiative"

*The only caveat with Darcs is that it's written in Haskell and hence has a few awkward (as in not-widely-installed) dependencies, but it is a small price to pay for the benefits it gives you.*

—Charles Goodwin, June 2004

## Conclusions

Haskell has worked well for darcs.

- There are plenty of existing Haskell coders out there who would love to contribute to a project written in Haskell.
- Haskell is pleasant language to learn, and experienced non-Haskell coders will learn it if they would like to contribute to your project.
- Optimization in Haskell is challenging, but possible.