The Secret History of Open Source Software Practices: Their Corporate and Scientific Origins, 1954-1980

Thomas Haigh

The Haigh Group/
University of Wisconsin, Milwaukee

UIUC, November 2007

Research supported by SIAM with funds from grant # DE-FG02-01ER25547 awarded by the US Department of Energy.

Structure of Talk

1. Review of canonical accounts of the origins of open source/free software
   - Linus Torvalds and Linux
   - Raymond Stallman and GNU
   - The Hacker Culture and Bell Labs
2. Examination of the role of the IBM SHARE scientific user group in the 1950s
   - Part of larger project on mathematical software
3. Mathematical Software in the 1970s
   - Hybrid of scientific publishing and commercial software industry
4. Some preliminary conclusions

1: Origins of Open Source Software – Three Fables

Open Source Idea?

- The **basic idea behind open source** is very simple: When programmers can read, redistribute, and modify the source code for a piece of software, the software evolves. People improve it, people adapt it, people fix bugs.

From OpenSource.org homepage

- "Open Source" concept attributed to 1998 meeting, Eric S. Raymond

Version 1: Finland, 1991

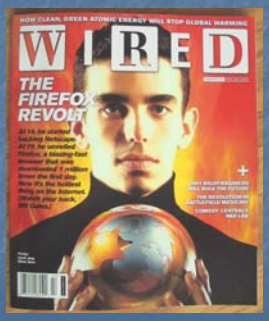
- Linus Torvalds sends a message to the comp.so.minix newsgroup…
- Linux was project of Linus Torvalds
  - Begun in 1991 as undergrad in Finland
- Now a leading server operating system

```
From:
torvalds@klaava.Helsinki.FI
(Linus Benedict Torvalds)
Newsgroups: comp.os.minix
Subject: Gcc-1.40 and a
posix-question
Message-ID:
<1991Jul3.100050.9886@klaava
.Helsinki.FI>
Date: 3 Jul 91 10:00:50 GMT

Hello netlanders,
Due to a project I'm working
on (in minix), I'm
interested in the posix
standard definition. Could
somebody please point me to
a (preferably)
machine-readable format of
the latest posix rules? Ftp-
sites would be
nice.
```

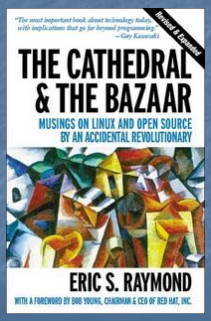

Power of the Internet

- Similar recent success for Firefox browser
- The story
  - Genius young programmer starts visionary project
  - Promising but incomplete versions posted on internet attract community of user/developers
  - A virtuous circle leads to exponential growth



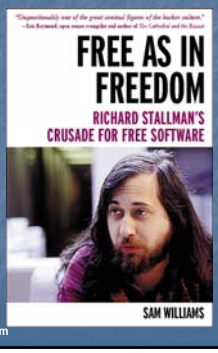
Bazaar Model

- Characteristics include
  - Users as co-developers
    - Projects start with personal problems to solve
    - Users debug systems – "many eyes make bugs shallow"
  - Early and frequent releases
  - High modularization
  - A "benevolent dictator" to lead project



Version 2: MIT, 1983

- Richard Stallman was respected MIT "hacker"
  - Author of EMACS editor
- Since 1984 Stallman Coordinates GNU project
  - GNU is Not Unix (recursive name)
  - Intended to produce open, free version of Unix
- "Free as in speech... not beer"

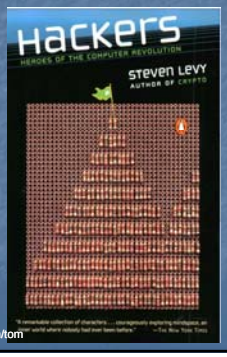


GNU's Free Software Definition

- The freedom to run the program, for any purpose (freedom 0).
- The freedom to study how the program works, and adapt it to your needs (freedom 1). Access to the source code is a precondition for this.
- The freedom to redistribute copies so you can help your neighbor (freedom 2).
- The freedom to improve the program, and release your improvements to the public, so that the whole community benefits (freedom 3). Access to the source code is a precondition for this.

Version 3: Hacker Culture

- Stallman was propagating and defending a tradition going back to the late 1950s at MIT
- Fundamentally oppositional
- Propagated and revitalized by
  - Personal computes
  - Widespread internet access



The Hacker Ethic

- Access to computers... unlimited and total
- All information should be free
- Mistrust authority – promote decentralization
- Hackers should be judged by their hacking...
- You can create beauty and art on a computer
- Computers can change your life for the better

From ch. 2 of Hackers, by Steven Levy, 1984

Summary of 3 Conventional Views

- Stress
  - Hacker culture and ideological commitments
  - Unpaid enthusiast virtuosos
  - Charismatic individuals
  - Novel licensing arrangements
- All about systems software
- All opositional
  - To commercial software
  - To official university culture

A New Origin Story for Many Open Source Practices

- Scientific software libraries
- 1950s
- No concern with licensing arrangements
- Claim to be motivated by pragmatic commercial interests
  - Avoidance of duplicated efforts on generic programs
  - To free up resources for areas of proprietary interests

2: Mathematical Software and Open Source

Scientific Computing

- Original function of early machines
  - Harvard Mark I, ENIAC
  - Source of the term "computer"
- Many applications are concerned with modeling natural or man made systems
  - Hydrogen bomb physics
  - Fluid Dynamics of air for aerospace
  - Celestial mechanics for space navigation

Mathematical Libraries

- Produced internally within computer centers
  - First example for EDSAC circa 1950
    - Invented along with subroutine
    - Discussed in 1951 programming text
    - Included Runge-Kutta differential equation routine
  - Routines stored on 5 track paper tape

Precursor: the Harvard Mark I (from Gerard Alberts) →

Issues - Mathematical

- Different numerical approximations suited to different problems
  - May be very slow
  - May give meaningless or inaccurate result
    - Problems may be under very specific conditions
- Newer, better methods may be more complex or highly specialized
  - Package in software for easy consumption
  - Disseminate formerly tacit knowledge between sites

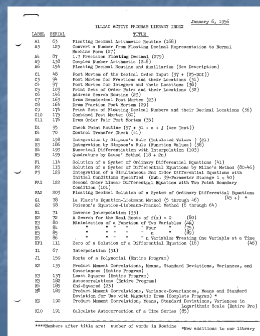
Early Needs

- Initially: very basic assembly language subroutines
  - Multiplication, square root, binary to decimal, floating point simulation, etc.
- FORTRAN (1956) covers basics, but plenty of challenges left
  - Each computer center is likely to need routines for
    - Linear algebra and matrix manipulation
    - Ordinary and Partial Differential Equation solvers
    - Special and Elementary functions
    - Curve fitting and least squares
    - Fast Fourier Transformation

Support for Library Work

- First US grant to support development may be for ILLIAC
  - Numerical Analysis funding from ONR 1950-1958

  Subroutine library 1955 →



January 6, 1956

ILLIAC ACTIVE PROGRAM LIBRARY INDEX

LABEL	SERIAL	TITLE
A1	63	Floating Decimal Arithmetic Routine (168)
A3	125	Convert a Number from Floating Decimal Representation to Normal Machine Form (27)
A4	87	1.7 Precision Floating Decimal (279)
A5	138	Complex Number Arithmetic (248)
A6	154	Floating Decimal Routine and Auxiliaries (See Description)
C1	48	Post Mortem of the Decimal Order Input (37 + (25-DOI))
C3	94	Post Mortem for Fractions and their Locations (31)
C4	97	Post Mortem for Integers and their Locations (38)
C5	103	Print Sets of Order Pairs and their Locations (32)
C6	146	Address Search Routine (23)
C7	163	Drum Sexadecimal Post Mortem (23)
C8	164	Drum Fraction Post Mortem (29)
C9	174	Print Sets of Floating Decimal Numbers and their Decimal Locations (36)
C10	175	Combined Post Mortem (80)
C11	176	Drum Order Pair Post Mortem (35)
D1	95	Check Point Routine (57 + 31 + s + j (see Text))
D4	70	Control Transfer Check (41)
E2	158	Integration by Simpson's Rule (Tabulated Values) (21)
E3	186	Integration by Simpson's Rule (Function Values) (38)
E4	193	Numerical Differentiation with Interpolation (103)
E5	195	Quadrature by Gauss' Method (18 + 2n)
F1	114	Solution of a System of Ordinary Differential Equations (41)
F2	115	Solution of a System of Differential Equations by Milne's Method (80-46)
F3	129	Integration of n Simultaneous 2nd Order Differential Equations with Initial Conditions Specified (Sub. 39-Parameter Storage 1 = 40)
FA1	122	Second Order Linear Differential Equation with Two Point Boundary Condition (101)
FA2	203	Floating Decimal Solution of a System of Ordinary Differential Equations (45 +) *
G1	78	La Place's Equation-Liebmann Method (5 through 46)
G2	98	Poisson's Equation-Liebmann-Frankel Method (6 through 64)
H1	71	Inverse Interpolation (33)
H2	72	A Search for the Real Roots of f(x) = 0 (80)
H3	80	Minimization of a Function of Two Variables (46)
H4	84	" " " " " Four " (75)
H5	85	" " " " " n " (89)
H6	86	" " " " " n Variables Treating One Variable at a Time (46)
HF1	111	Zero of a Solution of a Differential Equation (16)
I1	67	Interpolation (51)
J1	159	Roots of a Polynomial (Entire Program)
K2	135	Product Moment Correlations, Means, Standard Deviations, Variances, and Covariances (Entire Program)
K3	137	Least Squares (Entire Program)
K5	182	Autocorrelations (Entire Program)
K6	185	Chi-Squared (23)
K8	189	Product Moment Correlations, Variance-Covariances, Means and Standard Deviation for Use with Magnetic Drum (Complete Program) *
K9	190	Product Moment Correlation, Means, Standard Deviations, Variances in Logarithmic Scale (Entire Pro)
K10	191	Calculate Autocorrection of a Time Series (85)

****Numbers after title are: number of words in Routine   *New additions to our library

www.tomandmaria.com/tom

3: SHARE and Mathematical Software

www.tomandmaria.com/tom

IBM 701/704/709

- Large, "first generation" machines of 1950s
  - Worth approximately $2 million
- Designed for technical computation
  - Early users dominated by Southern California aerospace firms
  - Cold war context
- Many employees for each computer installation

704 at LLNL, 1956

www.tomandmaria.com/tom

Argonne Case Study

- Argonne National Laboratory (Yood dissertation topic)
  - Computer building starts 1949
  - 2 ENIAC women hired for first library in 1951
- IBM 704 arrives in 1957
  - Standard hardware
  - Still rely on internally developed library
- Applied Mathematics Division formed 1956
  - Consolidation of 50 staff members
  - Monopoly on electronic computing
  - Division seeks ability to support computing research (vs. service)
  - Repeated reorganizations

www.tomandmaria.com/tom

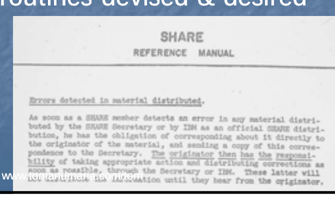
SHARE IBM User Group

- SHARE founded 1956
  - Cooperative group for users of large IBM computers
    - Discussions begin among IBM 701 users
    - SHARE represents "large" IBM scientific machine users
    - Representatives from each installation (52 by end of 1956)
      - Usually installation head or deputy
      - Engineering/science background, advanced degrees common
  - Intended to "share" programs, expertise, experiences and best practices
    - Lobbying of IBM to alter machines or policies

www.tomandmaria.com/tom

SHARE Software Library

- Routines contributed by user sites
  - Reproduction and catalog handled by IBM
  - Classification scheme developed to organize
  - Contributors responsible for maintenance
- List posted of routines devised & desired



SHARE
REFERENCE MANUAL

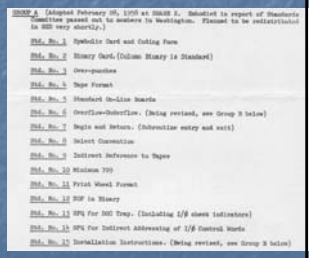
Errors detected in material distributed.

As soon as a SHARE member detects an error in any material distributed by the SHARE Secretary or by IBM as an official SHARE distribution, he has the obligation of corresponding about it directly to the originator of the material, and sending a copy of this correspondence to the Secretary. The originator then has the responsibility of taking appropriate action and distributing corrections as soon as possible, through the Secretary or IBM. These latter will ... information until they hear from the originator.

www.tomandmaria.com/tom

SHARE Practices

- Standardization needed to share code and practices
- Standardize machine configuration
  - Setting of switches, control panels, etc
- Standardize system software
  - Assembler and utility programs (not supplied by IBM)
  - Leads to big project to create "Share Operating System"



SSD

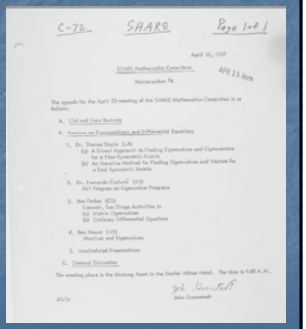
- Mechanism for communication between meetings
  - Mailing of large bundles of assorted materials
    - Committee reports
    - Drafts for comments
    - Letters, inquiries and responses
      - Including bug reports
- Also microfilms of source code for programs

SHARE Labor

- Installation reps are senior figures
  - Responsible for design and specification
  - Commit employees of their firms to develop code
- Economy of effort in developing generic routines
  - Driven by economics – save time and money
  - No proprietary advantage in cosine routine

SHARE Structure

- Committees to manage particular projects
  - Mathematical software is one important area
  - Subcommittees for particular projects



SHARE and the Four Freedoms

- Freedom to run – YES
- Freedom to study and adapt source code - YES
- Freedom to redistribute – YES
  - Pretty much all 704/9/90 were members
- Freedom to improve and release to the public – YES

Similarities in Practices

- Ad-hoc collaboration groups for specific projects
  - Some effort at modular code architecture
- Mechanisms to share and respond to bug reports
- Standards for coding and configuration to facilitate collaboration
- Open circulation of proposals and design documents
  - "Indoctrination" into culture

Challenges to SHARE

- Problems develop in open source model
- See Akera – "The Limits of Voluntarism", T&C, 2001
  - Following problems with the "SHARE Operating System" project the writing of system software migrates to IBM
- But mathematical software largely doesn't
  - SHARE is main distribution mechanism until early 1970s
    - Large labs rely on own code libraries

Packaging Expertise

- Craft knowledge of numerical methods formerly a part of carrying out computation
  - Held by generalist scientist/engineer, covered in textbooks
  - Intensive computation sometimes carried out by specialists
- Exchange of code spreads local practices beyond individual labs
  - Eventually leading to homogenization
- Code to solve specific equation types is now standardized and reused
  - Enables shift to newer, more complex mathematical methods

Black Boxing Expertise?

- In many ways, yes.
- But invocation of subroutines be dangerous without knowledge of methods used
  - May work very slowly or give meaningless results with specific equation
  - Library creators try to keep users aware of internal functioning – support role
- So is it a translucent box?

Immutable Mobile?

- Latour, Science in Action
  - Artifacts issued by "centers of calculation" to "act at a distance"
    - Associated with adoption of printing
  - Mobile (within & between labs)
  - Immutable (sometimes)
  - Presentable (yes)
  - Readable (yes – open source)
  - Combinable with each other (that's the point)
- Software seems to fit the description better than anything else!

3: Mathematical Software in the 1970s

Division of Labor

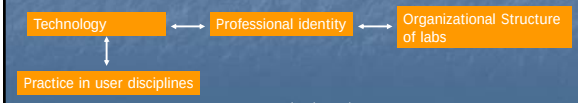
- Author of application programs may not be computer specialist
  - Writes outline code for specific task
  - Most of the work accomplished by subroutine calls to standard routines written by experts
- Shift supports new groups of methods specialists
  - Expertise encapsulated in code
  - Some sharing of codes between labs
- By early 1970s, emerging as discipline
  - Conferences
  - Books
  - Journals
  - Interest groups
- Situated between applied mathematics & computer science

Argonne Case II

- "Mathematical Algorithms Group" (20 people in late 1960s)
  - Distinct from "applied" and "systems" programming teams
  - Write, document new routines & improve old ones
  - Provide consulting to application programmers
  - Evaluate and modify externally produced routines
  - Argonne Code Center distributes routines
- 1970s: EISPACK (matrix routines) & LINPACK (linear algebra) projects
  - Collaboration with leading academic specialists
  - World class, portable packages in specialized areas

New Organizational Structures

- Computer departments provide new & secure location for expertise in applied mathematics
  - Library teams created in all(?) national labs
- Limitations of this position
  - Struggle to justify research agenda
  - Tend to collapse as computing is decentralized in 1980s
- Interplay between



Emergence of Research Community

- Spate of mathematical software activity in early 1970s
  - ACM SIGNUM has newsletter, conferences
  - Series of Mathematical Software conferences (John Rice of Purdue)
  - Creation of new journal
- Emergence of distinct research agenda
  - Mathematical software as unique field
  - Blending of applied mathematics with concerns from computer architecture and software engineering

Three Packaging Models

1. Peer Review and publication in journals
   - ACM TOMS
2. Commercial sale of software libraries
   - By IMSL and NAG
3. Creation of specialized packages by small teams of experts
   - E.g. LINPACK and EISPACK projects

- These models are not exclusive!
  - Same code can be made available in all three versions

1: Peer Review

- SHARE Numerical Analysis Project
- Attempt to peer review mathematical routines
  - Volunteer committee with IBM support
    - Hirondo Kuki
  - Limited success
    - Reviewing standards lacking and commitment uneven
  - Too many routines to review

ACM TOMs

- Transactions on Mathematical Software
- Publication venue for mathematical software
  - Started 1975 by John Rice
  - Program source code distributed via microfiche, card and tape
- Professional credit for programming accomplishments
  - "Algorithms" in Algol previously published in Communications of the ACM
  - And in Numerische Mathematik

2: Commercial Software Libraries

- NAG (UK) and IMSL (US)
- Comprehensive, commercial libraries
  - Both launched around 1972
  - Rapidly ported to multiple platforms
  - Numerical and statistical coverage
- Sold on annual subscription basis
  - Documented
  - Supported
  - Tested

3: The "PACK" Model

- EISPACK computes eigenvalues and eigenvectors of matrices
  - Released 1972
  - Standard routines in this area for a decade
- FORTRAN conversion of Algol routines by James H. Wilkinson and Christian Reinsch
  - Which in turn implemented new, dramatically improved methods
- Model widely adopted
  - Dozens of specialized packages produced within the labs during this era: FUNPACK, MUDPACK, FISHPACK etc.

EISPACK Development Methodology

- Grant funding received to test new methodology
- Very small team of contributors
  - Remains small for LINPACK follow-on project
- Debugging mostly done in small groups
  - Prior to release
  - Don't expect much insight from ordinary users
    - No expectation of code fix submission
  - Relationships cultivated with computer center staff
    - Create closed network of test sites
- Three major releases
  - Cycle repeated

Models not seen as opposed

- Many authors allow inclusion of code in all three types of package
  - EISPACK routines included in IMSL
- Pragmatic interest in getting code used
  - Salaries already paid by lab or university
  - No concern with copyright or licensing arrangements
  - Extension of social norms & practices of science
- Academic and commercial communities mixed
  - Ph.D.s work for library companies, their mentors sit on advisory boards
  - Some employees of library companies contribute to "PACK" projects

4: Concluding Ponderings

Commercial Origins of Open Source Practices in 1950s

- To recap, by 1956 we already have
  - All formal characteristics of "free" software
  - Many practices of modern open source development
- But not the ideology of free software
  - Seen as pragmatic actions, economically driven sharing

Hidden Commonality

- Shared engineering culture?
  - 1950s MIT Hackers
  - 1950s Aerospace engineering computing groups
- Seek to solve tasks in technically efficient manner
  - Avoid needless duplication of work
  - Provide tools to people who need them

www.tomandmaria.com/tom

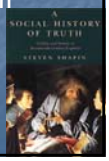
Richness of Models

- Commercial/Closed vs. Free/Open is
  - Recent dichotomy
  - Rhetorical construction
- Different communities produce many other models
  - Mathematical community starts with many corporate open source practices
  - Shifts to peer review and the elitist PACK model.

www.tomandmaria.com/tom

Shows need for Separation of Ideology and Practice

- Open source practices are older, more widespread than open source movement, so...
  - How important is the ideology?
  - Is selective use open source by big firms (IBM etc) the exception or the rule?
- How important are scientific norms to open source practices?
  - Publication and sharing of data
  - Goes back to 17th century gentlemen



www.tomandmaria.com/tom