The Efficiency of Open Source Software Development

Stefan Koch
Department of Management
Bogazici Üniversitesi
Istanbul

Koch, S. (2008) "Effort Modeling and Programmer Participation in Open Source Software Projects", *Information Economics and Policy*, 20(4): 345-355.

Contents

Introduction

...it would cost over $1 billion to develop...Red Hat 7.1...by conventional proprietary means...

David A. Wheeler (More Than a Gigabuck: Estimating GNU/Linux's Size, http://www.dwheeler.com/sloc/redhat71-v1/redhat71sloc.html, 2001)

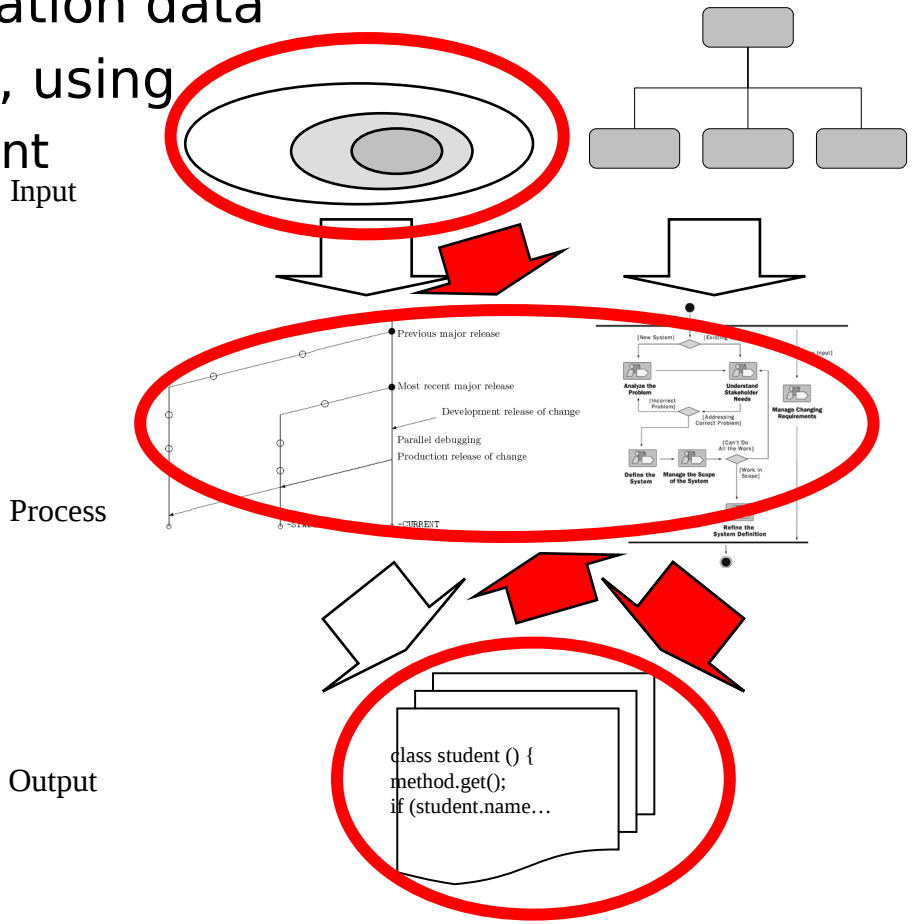
But, what did it really cost, and what does that mean?

Introduction

- Question of efficiency of open source development
  - efficiency denotes relation between input and output
  - in software development, input is effort, output is software
- How much software did we get for our effort?
  - More or less than investing the same effort in conventional development?
  - Is OS development a waste of resources?
  - Should we adopt the methodology for other areas?
- Discussion without much empirical basis
  - OS development is fast and cheap, provides high-quality
  - finding bugs late in life-cycle is inefficient
  - huge effort for OS development is spread and hidden
  - ... (e.g. IEEE Software 1999)

Introduction

- Idea: comparison between two estimated efforts
  - effort based on participation data
  - effort based on product, using
    - commercial development assumption
  - comparison



Methodology and Data Set

- Empirical research in open source
  - analysis needs to be based on quantitative information from a variety of project forms and sizes
  - case studies of well-known projects helpful
  - general assessment outstanding
- Mining software repositories
  - source code versioning systems, bug reporting systems, mailing lists,...
  - contain wealth of information on underlying software and associated development processes
    - cost-effective (no additional instrumentation necessary)
    - does not influence process
    - longitudinal data available
  - especially for OSS development (open, all information included)

Methodology and Data Set

- SourceForge.net
  - software development and hosting site, variety of services offered to hosted projects
  - statistics published (e.g. activity) not detailed enough
- Method
  - automatically extract data from web pages
  - generate CVS commands for download and logging
  - parsing of results
  - storage in a database (e.g. size, date, programmer of each checkin)
  - mostly using Perl scripts
  - analyses with statistical package

Methodology and Data Set

- Data from 8,621 projects
  - 7,734,082 commits
  - 663,801,121 LOCs added
  - 87,405,383 deleted
  - 2,474,175 single files
  - 12,395 distinct programmers
  - distribution of assets available, i.e. programmers, and resulting outcome very skewed
  - distribution of effort within projects very skewed (top decile --> 79% of code base, second decile --> 11%)

Participation-based Estimation

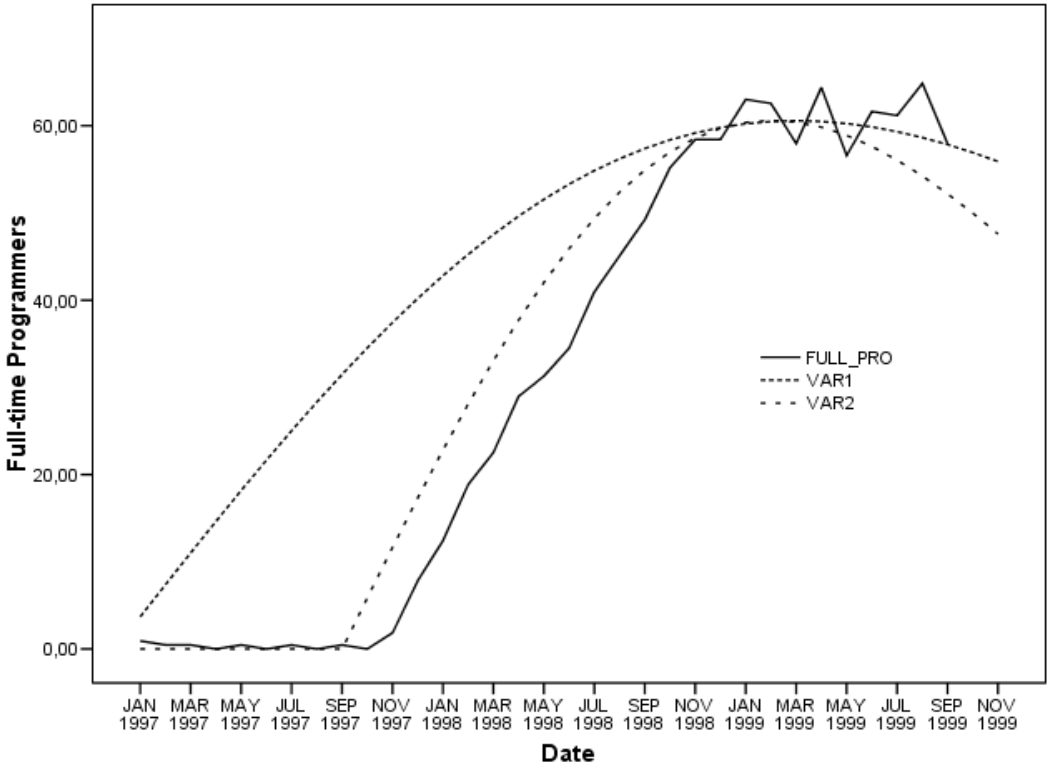
- Effort estimation based on actual participation
  - active programmers as base
  - number of persons with >0 commits in time interval (e.g. one month)
  - high correlation with LOC added in month
- Approaches
  - cumulation of active programmer months
  - manpower function modeling – function of people in project over time
- Conversion
  - necessary in both cases
  - open source person-month not equal commercial person-month
  - several studies, e.g. 18.4 hours per week spent on OSS development (Linux kernel developers)

Participation-based Estimation

- Manpower function modeling
  - Norden-Rayleigh model (first published 1960)
  - for R&D projects in general
  - project is a set of problems to be solved
  - number of problems is unknown but finite
  - finding and solving problems is accomplished by creative effort of applied manpower
  - event of solving a problem is independent and random (follows a Possion distribution)
  - number of persons gainfully employed in a project is dependent on the number of problems known but not solved
  - linear learning rate over time p(t)= 2 a t
  - m(t) = C‘(t) = 2 K a t exp(−a t^2)
  - Rayleigh-type curve (from 0 increasing up to peak point, where K can be computed, and to 0 again)

Participation-based Estimation

- Example: GNOME
  - 2 variants with different starting times
  - no decline visible
  - introduction of new requirements?



Participation-based Estimation

- Manpower function modeling revisited
  - include introduction of new requirements
  - several possible new functions proposed
    - number of additional problems introduced proportional to number of problems in the starting set
    - number of problems introduced proportional to number of problems already solved
    - with different learning rates
    - plus linear and quadratic model
    - overall 9 models
  - all fitted for Sourceforge.net projects (and 2 subsets) and compared one against the other (Wilcoxon signed-rank test with Bonferroni correction)
  - modified functions outperform, best is quadratic learning rate and proportional to starting set

Product-based Estimation

- Estimation models from commercial context
  - COCOMO 81 and COCOMO 2 (with different parameters)
  - function points (several different estimation formulae)
  - overall 6 different estimations per project
- Product estimated
  - measured in lines-of-code at end of observation
  - function points derived from lines-of-code (using backfiring – conversion rate per programming language)
  - current status seen as end product of commercial software development effort

Comparison and Results

- Comparison COCOMO – Norden-Rayleigh
  - analytically possible, Norden-Rayleigh used by COCOMO
  - find parameter values within COCOMO to generate Norden-Rayleigh curve from participation
  - for COCOMO 81 not possible, for COCOMO 2 very favourable parameters necessary
  - hints at high efficiency
- Comparison of all estimations
  - all estimation models applied to projects
  - note different time frames
  - product-based assumes end-product, as does cumulation
  - manpower function modeling does not

Comparison and Results

	N	Min.	Max.	Mean	Std. Dev.	Median	Sum over all projects
Norden-Rayleigh (equation-based)	8,620	0.06	200	0.69	3.72	0.19	5,965
Norden-Rayleigh (regression)	2,192	0.10	11,337.49	29.07	299.47	0.74	63,711
Norden-Rayleigh mod. equation (4)	944	-2,996.65	1,001.53	-117,62	210.23	0.11	-111,037
Norden-Rayleigh mod. equation (6)	944	-508.47	59,692.56	396,11	2,422.63	0.46	373,925
Norden-Rayleigh mod. equation (8)	649	-295,987.41	14,288.15	-2,197.98	21,352.57	0.35	-1,426,491
Norden-Rayleigh mod. equation (10)	1,743	-14.01	25.77	3.06	4.47	3.07	5,341
Norden-Rayleigh mod. equation (12)	755	-23.44	35.00	1.44	3.31	0.66	1,089
Active programmer effort years	8,620	0.10	55	0.56	1.39	0.12	2,229
COCOMO 81	8,621	0.00	4,159	18.56	133.66	2.02	160,020
COCOMO II (nominal parameters)	8,621	0.00	8,156	31.84	254.73	2.76	135,112
COCOMO II (realistic parameters)	8,621	0.00	3,539	15.67	113.55	1.69	274,482
Function Point (Albrecht and Gaffney)	8,621	-7.34	3,808	12.36	126.51	-4.67	106,528
Function Point (Kemerer)	8,621	-10.17	3,650	8.73	121.35	-7.61	75,249
Function Point (Matson et al., linear)	8,621	0.32	1,069	5.84	35.42	1.07	50,316
Function Point (Matson et al., log.)	8,559	0.00	869	4.52	28.93	0.62	38,695

Comparison and Results

- Results
  - series of Wilcoxon signed-rank tests (also 2 subsets)
  - Norden-Rayleigh significantly below all product-based estimates
  - cumulation significantly below (but different time frame)
- Interpretation
  - OS development more efficient by using self-selection for task management etc.
  - extremely high amount of non-programmer participation (1:7 relation, ‘chief programmer team’ organisation)

Conclusions and Future Work

- Efficiency of open source development
  - first results promising
  - but several explanations (efficiency, non-programmer participation and special organisation,...)
  - effort estimation problematic, development of new models necessary
  - effort logging of participants would be interesting
  - significance of results for method adoption in both directions
  - developers would not abandon OS development because of different motivations, but do not want to waste their time either

Conclusions and Future Work

- Future Work
  - inclusion of quality indicators
  - application of Data Envelopment Analysis (DEA)
    - non-parametric efficiency comparison method
    - multi-input, multi-output systems with factors on different scales
    - possibility to include different input factors (programmers, posters, commercial sponsorship,...) and output factors (size, quality, downloads,...)
    - compare OSS projects for influences on efficiency (inequality, licence, tools,...)
    - compare OSS with closed-source projects