Why is Software Cost Estimation so Hard at NASA?

Tim Menzies
Lane Department of Computer Science
West Virginia University
tim@menzies.us

Jairus Hihn
Jet Propulsion Laboratory/ California Institute of Technology
jhihn@jpl.nasa.gov

Karen Lum
Jet Propulsion Laboratory/ California Institute of Technology
ktlum@jpl.nasa.gov

NASA Cost Symposium
Cleveland, Ohio
June 20-22, 2006

Sound Bites

- Don’t assume ‘it’ works: Check ‘it’ locally
- Too many cost drivers
  – Can’t justify because …
- … Large variance problem
- No more cherry picking
  – We can use more data

Introduction

- The NASA Office of Safety and Mission and Assurance funds a number of research initiatives to improve software reliability
  - They are also interested in improving their own capability to estimate the level of IV&V resources that should be allocated to each NASA mission
  - The result was that OSMA was willing to find a small research effort to provide them with the data and models they wanted while extending the state of the art in software cost estimation using data mining techniques
- Today we will report on findings from analyzing a NASA COCOMO 81 dataset with 93 records. (Paper published in proceedings of ISPA 2006 Conference where it won best paper in Software Track)
- Our current tool is called COSEEKMO
  - This methodology can be applied to any set of cost models and data (Hardware, Software, Systems, Mission, Instrument, Commercial)
  - COSEEKMO was developed because we had access to a fairly large COCOMO data set.
  - We are also analyzing proprietary COCOMO II data sets

Local Calibration (cont.)

COCOMO LC procedure applied to stratifications of data from three software repositories*

Line	Source	Subset	Number of examples (a)	Mean error (b) (%)	b/a (%)
	Coc81	All	63	42	100
1	Coc81	Kind = max	31	47	111
2	Coc81	Mode = embedded	28	42	100
3	Coc81	Lang = fortran	24	50	119
4	Coc81	Mode = organic	23	32	76
5	Coc81	Kind = min	21	47	111
6	Coc81	Lang.mol	20	34	80
	CocII	All	161	20	100
7	CocII	DevelopmentEnd = 1998	54	11	55
8	CocII	Organization = 2	48	19	95
9	CocII	DevelopmentEnd = 1980	48	19	95
10	CocII	Dev.waterfall	48	27	135
11	CocII	Organization = 1	34	9	45
12	CocII	DevelopmentEnd = 1991	22	27	135
13	CocII	Language = C	21	23	115
	Nasa93	All	93	60	100
14	Nasa93	Sub5	80	53	88
15	Nasa93	Mode = semiDetached	69	58	96
16	Nasa93	Sub4	39	80	133
17	Nasa93	Sub9	38	81	135
18	Nasa93	Sub7	38	68	113
19	Nasa93	Sub8	37	82	136
20	Nasa93	Sub3	37	43	71
21	Nasa93	Sub1	30	43	71
22	Nasa93	Sub6	23	56	93
23	Nasa93	Mode = embedded	21	188	313
24	Nasa93	Sub2	20	46	76

b/a = 100%

*Results in green show where standard practice improved cost estimation; results in red show where standard practice made the models worse.

COSEEKMO

- COSEEKMO is a tool that derives effort estimation models from COCOMO data sets
  - Standard and non standard models
  - Basic approach can be generalized but we only had COCOMO 81 and COCOMO II data to work with
- COSEEKMO performs an exhaustive search over all parameters and records in order to guide data pruning
  - Records (Stratification)
  - Variables (Wrapper)
- COSEEKMO uses Different Calibration and Validation Datasets
- COSEEKMO measures model performance by multiple measures
  - Pred(30) - Number of actuals within +/- 30% of model estimate
  - MMRE - mean magnitude of relative error
  - R^2
  - Variance computed from parameter values and model performance across multiple derived models and performance against hold out data not standard regression computations. This yields different answers.
- COSEEKMO can be used to address the following questions

Data

- COSEEKMO built effort estimators using all or some part of two COCOMO 81 data sets (nasa93 and coc81). Each part selected some subset of the total records.
  - NASA93 consists of 93 flight and ground records form multiple NASA Centers that completed from the late 1970's through the late 1980's

Data	*Coc81:*	has 63 records in the COCOMO 81 format
	Nasa93:	has 93 NASA records in the COCOMO 81 format
Subsets/Stratification Categories	*All:*	selects all records from a particular source; e.g.. "coc81_all" and "nasa93_all"
	Category:	is a NASA-specific designation selecting the type of project; e.g. avionics, data capture, etc.
	Fg:	selects either "f" (flight) of "g" (ground) software
	Kind:	selects records relating to the development platform; max = mainframe and mic = microprocessor
	Lang:	selects records about different development languages
	Center:	*nasa93* designation selecting records relating to where the software was built
	Project:	*nasa93* designation selecting records relating to the name of the project
	Mode:	selects records relating to different COCOMO 81 development modes; *org* , *sd* , and *e* are short for organic, semi-detached, and embedded (respectively)
	Type:	selects different COCOMO 81 designations and include "bus" (for business application) or "sys" (for system software)
	Year:	is a *nasa93* term that selects the development years, grouped into units of five; e.g. 1970, 1971, 1972, 1973, 1974 are labeled "1970"

Survivors from Rejection Rules

		Records		Treatment			Results		
row	source:part	T=*|train|*	T=*|test|*	Numbers	*|Subset|*	Learn	Mean PRED(30)	MMRE mean	MMRE Sd
1.	coc81:kind.min	11	10	precise	17	e	60	31	21
2.	coc81:lang.ftn	14	10	precise	17	sd	42	44	30
3.	coc81:mode.e	18	10	precise	17	e	46	40	34
4.	coc81:kind.max	21	10	precise	17	e	52	38	33
5.	coc81:all	53	10	precise	17	LC	50	40	37
6.	coc81:mode.org	13	10	precise	17	org	62	32	33
7.	coc81:lang.mol	10	10	precise	17	sd	56	36	41
8.	nasa93:project.Y	13	10	precise	16	LC	78	22	20
9.	nasa93:category.missionplanning	10	10	rounded	17	e	50	36	37
10.	nasa93:category.avionicsmonitoring	20	10	precise	8	M5P	53	38	39
11.	nasa93:mode.sd	59	10	rounded	7	LC	62	33	34
12.	nasa93:project.X	28	10	precise	17	e	42	42	45
13.	nasa93:fg.g	70	10	rounded	10	LSR	65	32	39
14.	nasa93:center.5	29	10	precise	12	LC	43	57	70
15.	nasa93:year.1975	27	10	precise	11	LSR	52	50	62
16.	nasa93:all	83	10	rounded	14	LSR	43	48	62
17.	nasa93:year.1980	28	10	precise	16	LC	53	53	80
18.	nasa93:mode.e	11	10	precise	17	e	42	64	100
19.	nasa93:center.2	27	10	precise	17	LC	83	22	38

Some Good News

- Physical SLOC always loads as significant with no language adjustment
- The standard functional form shown below is virtually always selected as indicated by the non-standard model M5P being selected only once

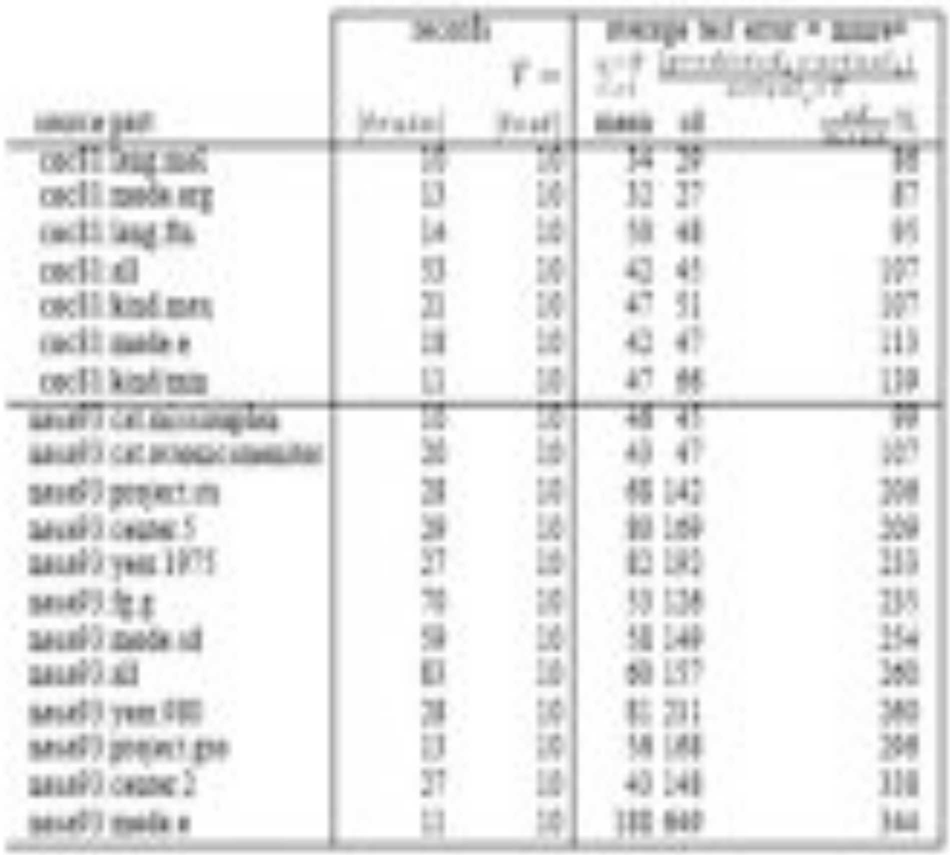
$$effort(personmonths) = a * \left(KLOC^{b}\right) * \left(\prod_{j} EM_{j}\right)$$

- Based on Books work need to study what he calls the triad

$$effort(personmonths) = c + a * \left(KLOC^{b}\right) * \left(\prod_{j} EM_{j}\right)$$

- The 'out-of-the-box ' version of COCOMO 81 is almost always the best model on the original COCOMO81 data
  - View as a sanity check on our methodology
- However, for the NASA93 data sometimes
  - one can use the model right out of the box
  - sometimes local calibration is sufficient
  - sometimes a full regression analysis needs to be performed to obtain optimal results

The Large Variance Problem



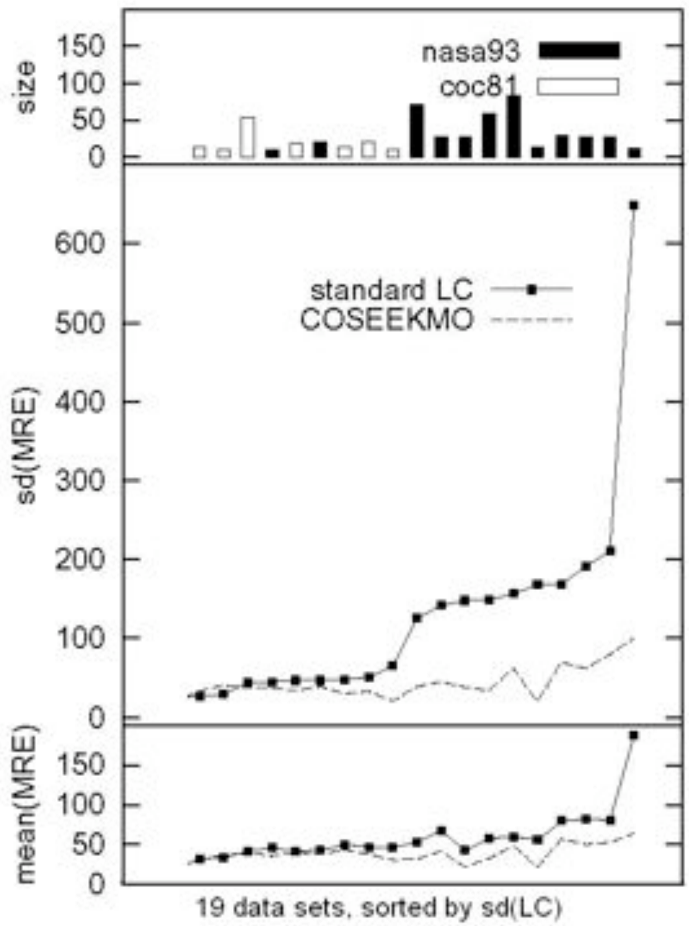
source part	|Train|	|Test|	mean	sd	[illegible] %
coc81 lang.mol	18	10	34	29	98
coc81 mode.org	13	10	32	27	87
coc81 lang.ftn	14	10	50	48	95
coc81 all	53	10	42	45	107
coc81 kind.max	21	10	47	51	107
coc81 mode.e	18	10	42	47	113
coc81 kind.min	13	10	47	66	139
nasa93 cat.[illegible]	18	10	48	47	99
nasa93 cat.[illegible]	26	10	43	47	107
nasa93 project.[illegible]	28	10	68	142	208
nasa93 center.5	29	10	80	169	209
nasa93 year.1975	27	10	82	182	213
nasa93 fg.g	70	10	53	128	235
nasa93 mode.sd	59	10	58	149	254
nasa93 all	83	10	60	157	260
nasa93 year.[illegible]	28	10	81	211	260
nasa93 project.[illegible]	13	10	58	168	290
nasa93 center.2	27	10	40	148	318
nasa93 mode.e	13	10	188	840	344

- The large variance problem is the most fundamental problem in cost estimation
- Causes our models to be unstable and brittle
- The COCOMO81 data has smaller variance but variance is still large and the data was 'worked'

- The average deviation on the error can grow to over 300 times larger than the mean

Local Calibration

Does Not Always Improve Performance



- For the NASA data set Local Calibration (LC) or re-estimating a and b only does not produce the 'best' model.
- A more thorough analysis is required including reducing the number of variables

- **Effort models were learned via either standard LC or COSEEKMO**
- **The top plot shows the number of projects in 27 subsets of our two data sources**
- **The middle and bottom plots show the standard deviation and mean in performance error**
- **Data subsets are sorted by the error's standard deviation**

Cost Driver Instability

Data Subset	COCOMO 81 Cost Drivers: acap	time	cplx	aexp	virt	data	turn	rely	stor	lexp	pcap	modp	vexp	sced	tool	Number of Significant Cost Drivers
coc81_all	❍	●	●	●	●	●	●	●	●	●	●	●	●	●	●	15
coc81_mode_embedded	❍	●	❍	❍	●	❍	❍	❍	❍	●		●	●	●	●	14
coc81_mode_organic	●	●	❍	●	●	●	●		❍		●	●	●	●	●	13
nasa93_all	●	●		●	●	●	●	●	●							8
nasa93_mode_embedded	❍	●	●		●	●	●	●	●	❍	❍			●		11
nasa93_mode_semidetached	●			●									❍			3
nasa93_fg_ground	●		❍	●						●	❍					5
nasa93_category_missionplanning	❍	●	●			●	●	●			●	❍		❍		9
nasa93_category_avionicsmonitoring	●			❍								●	❍	❍	❍	6
nasa93_year_1975	●	●	●	●	●	●		●	●	❍	❍					10
nasa93_year_1980	●	●	●	❍	●	●	●	●	●					●	❍	11
nasa93_center2	●	●	●	●	●	❍	●	❍	●	●	●	●	●		●	14
nasa93_center5		●	●	❍	●	●	❍	●	●	❍						9
nasa93_project_gro	❍	❍	●	❍	●		●	❍	❍	●	❍	●	●		❍	13
nasa93_project_sts		●	●		●	●	●	●	●							7
Usually Significant	5	1	3	5	0	2	2	3	3	3	4	1	2	2	3	
Always Significant	8	11	9	7	11	9	9	8	8	5	4	6	5	5	4	
Total Number of Significant Occurrences	**13**	**12**	**12**	**12**	**11**	**11**	**11**	**11**	**11**	**8**	**8**	**7**	**7**	**7**	**7**	

Legend:

● = Not significantly different than 10 at a 95% Confidence Interval

❍ = Not significantly different than 9 or greater at a 95% Confidence Interval

The bottom line is that we have way too many cost drivers in our models!

- Furthermore, what smaller set is best varies across different domains and stratifications
- The cost drivers that are unlikely to improve model performance are pcap, vexp, lexp, modp, tool, sced
- It is expected for more contemporary data that stor and time would drop out because there are fewer computer constraints these days and modp may become more significant

Sound Bites

- Don’t assume ‘it’ works: Check ‘it’ locally
- Too many cost drivers
  - Can’t justify because …
- … Large variance problem
- No more cherry picking
  - We can use more data
- Please, more repeatable studies and analysis
  - http://unbox.org/wisp/trunk/cocomo/data

Conclusion

- Our research indicates that
  - We can dramatically reduce the deviation in model performance
  - most cost models have far too many cost drivers.
  - No one model is best all of the time
- At a minimum COSEEKMO provides a way to fully analyze the properties of our models and more accurately determine cost estimation uncertainty
- Cost estimation uncertainty is measured more accurately when derived form model performance against a test set or hold out data set.
  - In general the estimation uncertainty will be larger then currently indicated by standard regression results

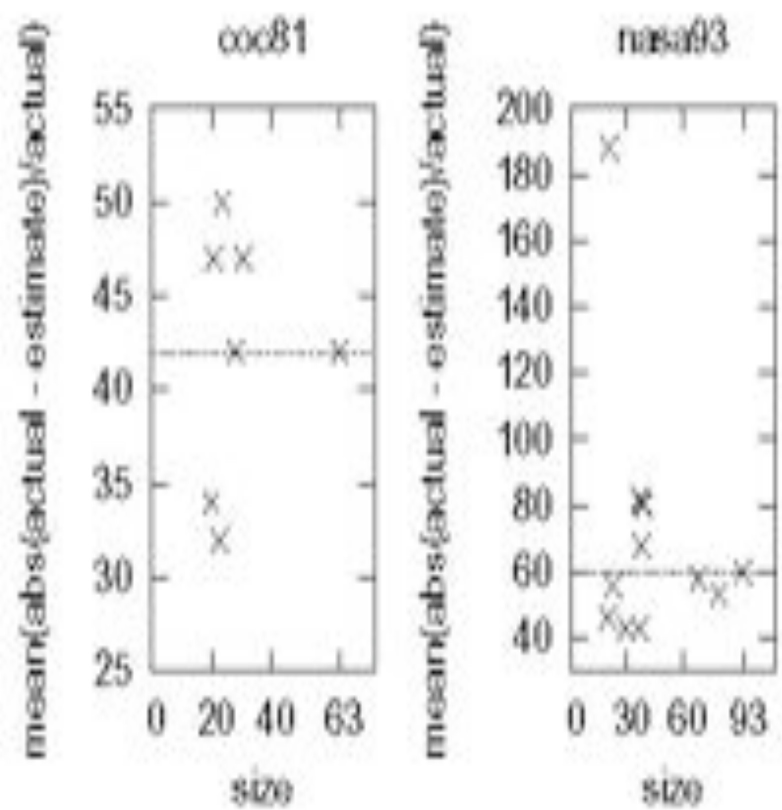
Open Source Data and Tools

- PROMISE repository of software engineering data sets
- COCOMO 81 (If too lazy to type it in):
  - http://promise.site.uottawa.ca/SERepository/datasets/cocomo81.arff
- COCOMO 81 NASA94:
  - http://promise.site.uottawa.ca/SERepository/datasets/cocomonasa_v1.arff
  - Ground mission support software from 70's to mid-80's
- Forthcoming
  - Add historical NASA flight records from 70's to mid-80's
  - COSEEKMO on-line
  - Feature Subset Selection Tool
    - Google for WEKA to obtain original research software

Key Questions in Cost Model Development

- What is a models real estimation uncertainty?
- How many records required to calibrate?
  - Answers have varied from 10-20 just for intercept and slope
  - If we do not have enough data what is the impact on model uncertainty
- Data is expensive to collect and maintain so want to keep cost drivers and effort multipliers as few as possible
  - But what are the right ones?
  - When should we build domain specific models?
- What are the best functional forms?
- What are the best ways to tune/calibrate a model?

Stratification
Does Not Always Improve Performance



- Stratification does not always improve model performance
- Results show it is 50-50
- Main implication is that ome must really know their data as there is no solution to determine the best approach to model calibaration

- The plots show mean performance error (i.e. |(predicted − actual)|/actual) found after 30 experiments with each subset
- The dashed horizontal lines shows the error rate of models learned from all data from the two sources
- The crosses show the mean error performance seen in models learned from subsets of that data
- Crosses below/above the lines indicate models performing better/worse (respectively) than models built from all the data