Predicting Fault Incidence Using Software Change History

Jing Huang

Background(1)

- **Code Decay**
  - structure tends to degrade
  - difficult to understand and change

- **purpose**

  To identify those aspects of the code and its change history that are most closely related to the numbers of fault

Background (2)

- **Product metrics**

  To be computed using syntactic data taken from a snapshot of the software
  - numbers of lines of code
  - degree of statement nesting
  - code complexity

Background (3)

- **Process Measures**

  To be computed using data taken from the change and defect history of the program
  - Number of past faults
  - Number of deltas to a module over its entire history
  - the average age of the lines
  - The development organization
  - the number of different developers
  - the extent to which a module is connected to other modules
  - A weighted time damp model

Research data

- Researched system

  a 1.5 million line subsystem of a telephone switching system

- Data sources
  - A initial Modification Request (IMR) database

    records of problems to be solved
  - delta database

    records for changes related to single files

Statistical models(1)

- stable model
  - To assume that the fault generation dynamics remain stable across modules over time
  - To predict numbers of future faults using numbers of past
  - To serve as a yardstick against which to compare other models

Statistical models(1)

- stable model
  - To assume that the fault generation dynamics remain stable across modules over time
  - To predict numbers of future faults using numbers of past
  - To serve as a yardstick against which to compare other models

Statistical models(2)

- Generalized Linear Models
  - To used a logarithmic link and took the error distribution to be Poisson
  - Three different intercepts for international, domestic, and common modules

Model	Intcp	Common	Intl	US	Error
(A) Stable	-	-	-	-	757.4
(B) Null model	-	-	-	-	3108.8
(C) Organization only	3.46	0	-0.13	-1.39	2587.7
(D) 0.84 log(lines/1000)	0.92	0	0.17	-0.92	1271.4
(E) −0.14 log(lines/1000) + 1.19 log(deltas/1000)	3.31	0	0.46	-0.70	980.0
(F) 1.05 log(deltas/1000)	2.95	0	0.43	-0.72	985.1
(G) 0.07 log(lines/1000) + 0.95 log(deltas/1000) - 0.44age	2.63	0	0.73	-0.65	696.3
(H) 1.02 log(deltas/1000) - 0.44age	2.87	0	0.74	-0.63	697.4

Statistical models(3)

- complexity metrics
  - nearly all of the complexity measures were virtually perfectly predictable from lines of code

TABLE 2
Correlations of Complexity Metrics

	1	2	3	4	5	6	7	8	9	10	11	12
1 Lines Of Code	1	.97	.88	.88	.91	.99	.98	.92	.97	.85	.72	.35
2 McCabe V(G)1	.97	1	.88	.90	.88	.95	.95	.89	.93	.86	.76	.29
3 Functions	.88	.88	1	.82	.89	.85	.84	.91	.84	.76	.65	.29
4 Breaks	.88	.90	.82	1	.83	.86	.85	.85	.85	.78	.67	.27
5 Unique Operators	.91	.88	.89	.83	1	.89	.87	1.00	.94	.65	.47	.48
6 Total Operands	.99	.95	.85	.86	.89	1	1.00	.90	.98	.85	.72	.31
7 Program Volume	.98	.95	.84	.85	.87	1.00	1	.88	.97	.87	.74	.28
8 Expected Length	.92	.89	.91	.85	1.00	.90	.88	1	.94	.69	.53	.42
9 Variable Count	.97	.93	.84	.85	.94	.98	.97	.94	1	.77	.60	.38
10 MaxSpan	.85	.86	.76	.78	.65	.85	.87	.69	.77	1	.92	-0.10
11 MeanSpan	.72	.76	.65	.67	.47	.72	.74	.53	.60	.92	1	-0.25
12 Prog Level	.35	.29	.29	.27	.48	.31	.28	.42	.38	-0.10	-0.25	1

Statistical models(4)

- Weighted Time Damp Model

  To estimate a module's fault potential by adding an explicit contribution from each MR to the module

$$e_i = \sum_{m=1}^{M} e^{-\alpha(t-T_m)} w_{im} \propto \sum_{m=1}^{M} e^{\alpha T_m} w_{im},$$

e_i :the fault potentials for the ith modules i=1..80

T_m :the time of the mth MR

t :current time

w_{im} :the weight to the ith module corresponding to the mth MR

1if mth MR touches ith module; 0 otherwise

number of lines changed as part of this MR

log(number of lines changed as part of this MR)

α :governs the rate at which the contribution of old MRs to the fault potential disappears

Conclusion

- the best predicted model

  The weighted time damp model

  predicted fault potential using a sum of contributions from all the changes to the module in its history (large and recent changes, deltas contribute the most to fault potential)

- The best generalized linear model

  The model which uses numbers of changes to the module in the past together with a measure of the module's age.

- poor predicted model
  - numbers of lines of code
  - degree of statement nesting
  - code complexity
  - the number of different developers
  - the extent to which a module is connected to other modules

Likes and Dislikes

- Likes

  To provide some ideas based on change history of software to predict bugs

  To develop several statistical models to evaluate the prediction of bugs

- Dislikes

  To use too many statistical formulas and most of them are very complicated

  The explanation about purpose to use some statistic formulas is not clear

Thank you!

Questions?