Predicting Fault Incidence Using Software Change History

Jing Huang
Background

- 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
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
**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
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
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 use a logarithmic link and took the error distribution to be Poisson
  - Three different intercepts for international, domestic, and common modules

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

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

<table>
<thead>
<tr>
<th>TABLE 2</th>
<th>Correlations of Complexity Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1 Lines Of Code</td>
<td>1</td>
</tr>
<tr>
<td>2 McCabe V(G)1</td>
<td>.97</td>
</tr>
<tr>
<td>3 Functions</td>
<td>.88</td>
</tr>
<tr>
<td>4 Breaks</td>
<td>.88</td>
</tr>
<tr>
<td>5 Unique Operators</td>
<td>.91</td>
</tr>
<tr>
<td>6 Total Operands</td>
<td>.98</td>
</tr>
<tr>
<td>7 Program Volume</td>
<td>.98</td>
</tr>
<tr>
<td>8 Expected Length</td>
<td>.92</td>
</tr>
<tr>
<td>9 Variable Count</td>
<td>.97</td>
</tr>
<tr>
<td>10 MaxSpan</td>
<td>.85</td>
</tr>
<tr>
<td>11 MeanSpan</td>
<td>.72</td>
</tr>
<tr>
<td>12 Prog Level</td>
<td>.35</td>
</tr>
</tbody>
</table>
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 \( i \)th modules \( i=1..80 \)
- \( T_m \): the time of the \( m \)th MR
- \( t \): current time
- \( w_{im} \): the weight to the \( i \)th module corresponding to the \( m \)th MR
  1 if \( m \)th MR touches \( i \)th module; 0 otherwise
- \( \alpha \): number of lines changed as part of this MR
- \( \alpha \): log(number of lines changed as part of this MR)
- \( \alpha \): 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?