Sampling & Metrics
Survey & MSR

• Many software repositories are available (Source code, bugs, tests, requirements)
• Why not using them to perform a survey?
  – Goal, Null hypothesis, operation, analysis
Example of Survey

Are library migrations frequent?

Cédric

Repository 1  Repository 2  Repository 3  Repository 4  ...  Repository n
Harmony: Browse Model

Cédric's Harmony analysis

4% of the studied repositories performed at least one migration

Repository 1
Repository 2
Repository 3
Repository 4
Repository n

18/06/2013
Issues

• Sampling
  – The more repositories the best the result of the survey?
  – Which repositories (big, old, small, young, active, with large community, …)?
  – How many repositories? How much of their resources?

• Metrics
  – Re-using existing measures?
  – How to define new measures?
Principles of statistics

• Individual (Unit of interest)
  – Object to measure (class, project, developer, …)
• The distributivity of the measures has an impact on the number of individuals to measure
  – Normal law ≈ 30 individuals
  – Flip a coin. With 10%. With 10 tests.
    • $P(1\ A\ and\ 9\ B) = P(1/10) = 0.0097$
    • $P(2/10\ or\ less) = 0.097$
    • $P(3/10\ or\ less) = 0.449$
Sampling - Principes

Primary unit selection has a major impact
Cluster Sampling

Primary units are selected until the sample is full

All individuals are selected

Requested sample size = 8, effective sample size = 9
Double Sampling

Sample of primary units

Smaller population of individuals

Sample of individuals

Requested and effective sample size = 8
Example on NM+NA

• NM + NA = Number of methods + Number of attributes
• 80/100 quantile
• 3 large projects
  – NM+NA = 27 (but no trust !)
• Double Sampling on GitHub (400 projects) + Bootstrap on 1000 classes
  – Nm+NA = [23-27] with 95% of confidence
Metrics

• Measure the quality of software development
• Analyse the impact on maintenance
• Identify anti-patterns
• Examples
  – Source Code: LOC, CBO, DIT, DIT, ...
  – Workload: NbCommits, Touches, CHURN
  – Bugs: NbOfOpenBug, TimeToFix
  – ...

18/06/2013
Metrics, effects and aggregation

- A metric should represent something
- Correlation to measure the « effect » of the metrics
- Aggregation of metrics
Detecting Fault Prone Component

- Use metrics to identify fault prone components
- Focus maintenance on these components

Figure 1. After mapping historical failures to entities, we can use their complexity metrics to predict failures of new entities.
Correlation

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
<th>Max</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Module metrics</strong></td>
<td>correlation with metric in a module M</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Classes</td>
<td># Classes in M</td>
<td>0.531</td>
<td>0.612</td>
<td>0.713</td>
<td>0.066</td>
<td>0.438</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td># Functions in M</td>
<td>0.131</td>
<td>0.699</td>
<td>0.761</td>
<td>0.104</td>
<td>0.531</td>
<td></td>
</tr>
<tr>
<td>GlobalVariables</td>
<td># global variables in M</td>
<td>0.023</td>
<td>0.664</td>
<td>0.695</td>
<td>0.108</td>
<td>0.460</td>
<td></td>
</tr>
<tr>
<td><strong>Per-function metrics</strong></td>
<td>correlation with maximum and sum of metric across all functions f() in a module M</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lines</td>
<td># executable lines in f()</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameters</td>
<td># parameters in f()</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Arcs</td>
<td># arcs in f()’s control flow graph</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Blocks</td>
<td># basic blocks in f()’s control flow graph</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ReadCoupling</td>
<td># global variables read in f()</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WriteCoupling</td>
<td># global variables written in f()</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AddrTakenCoupling</td>
<td># global variables whose address is taken in f()</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ProcCoupling</td>
<td># functions that access a global variable written in f()</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FanIn</td>
<td># functions calling f()</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Depends on both the metrics and software.
Combine Metrics

Table 5. Regression models and their explanatory power

<table>
<thead>
<tr>
<th>Project</th>
<th>Number of principal components</th>
<th>% cumulative variance explained</th>
<th>$R^2$</th>
<th>Adjusted $R^2$</th>
<th>F - test</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>9</td>
<td>95.33</td>
<td>0.741</td>
<td>0.612</td>
<td>5.731, p &lt; 0.001</td>
</tr>
<tr>
<td>B</td>
<td>6</td>
<td>96.13</td>
<td>0.779</td>
<td>0.684</td>
<td>8.215, p &lt; 0.001</td>
</tr>
<tr>
<td>C</td>
<td>7</td>
<td>95.34</td>
<td>0.579</td>
<td>0.416</td>
<td>3.541, p &lt; 0.005</td>
</tr>
<tr>
<td>D</td>
<td>7</td>
<td>96.44</td>
<td>0.684</td>
<td>0.440</td>
<td>2.794, p &lt; 0.077</td>
</tr>
<tr>
<td>E</td>
<td>5</td>
<td>96.33</td>
<td>0.919</td>
<td>0.882</td>
<td>24.823, p &lt; 0.0005</td>
</tr>
</tbody>
</table>
Predictors are accurate only when obtained from the same or similar projects.

Table 6. Predictive power of the regression models in random split experiments

<table>
<thead>
<tr>
<th>Project</th>
<th>Correlation type</th>
<th>Random split 1</th>
<th>Random split 2</th>
<th>Random split 3</th>
<th>Random split 4</th>
<th>Random split 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Pearson</td>
<td>0.480</td>
<td>0.327</td>
<td>0.725</td>
<td>-0.381</td>
<td>0.637</td>
</tr>
<tr>
<td></td>
<td>Spearman</td>
<td>0.238</td>
<td>0.185</td>
<td>0.693</td>
<td>-0.602</td>
<td>0.422</td>
</tr>
<tr>
<td>B</td>
<td>Pearson</td>
<td>-0.173</td>
<td>0.410</td>
<td>0.181</td>
<td>0.939</td>
<td>0.227</td>
</tr>
<tr>
<td></td>
<td>Spearman</td>
<td>-0.055</td>
<td>0.054</td>
<td>0.318</td>
<td>0.906</td>
<td>0.218</td>
</tr>
<tr>
<td>C</td>
<td>Pearson</td>
<td>0.559</td>
<td>-0.539</td>
<td>-0.190</td>
<td>0.495</td>
<td>-0.060</td>
</tr>
<tr>
<td></td>
<td>Spearman</td>
<td>0.445</td>
<td>-0.165</td>
<td>0.050</td>
<td>0.190</td>
<td>0.082</td>
</tr>
<tr>
<td>D</td>
<td>Pearson</td>
<td>0.572</td>
<td>0.845</td>
<td>0.522</td>
<td>0.266</td>
<td>0.419</td>
</tr>
<tr>
<td></td>
<td>Spearman</td>
<td>0.617</td>
<td>0.828</td>
<td>0.494</td>
<td>0.494</td>
<td>0.494</td>
</tr>
<tr>
<td>E</td>
<td>Pearson</td>
<td>-0.711</td>
<td>0.976</td>
<td>-0.818</td>
<td>0.418</td>
<td>0.007</td>
</tr>
<tr>
<td></td>
<td>Spearman</td>
<td>-0.759</td>
<td>0.577</td>
<td>-0.883</td>
<td>0.120</td>
<td>0.152</td>
</tr>
</tbody>
</table>
Properties [CK94]

- **Noncoaseness**
  - For each P there exists Q such that m(P) ≠ m(Q)

- **Nonuniqueness**
  - There can exist distinct classes P and Q such that m(P) = m(Q)

- **Design Details are important**
  - Given P and Q, which are similar, this does not imply that m(P)=m(Q)

- **Monotonicity**
  - Given P and Q, m(P) ≤ m(P+Q), m(Q) ≤ m(P+Q)

- **Noequivalence of Interaction**
  - Given P, Q and R, m(P) = m(Q), does not imply that m(P+R) = m(Q+R)

- **Interaction increases complexity**
  - m(P)+m(Q) < m(P+Q)
Conclusion

• Use existing data (OSS Repository) to perform survey
• Statistics and sampling
• Metrics and correlation