ICSE’14 Workshop Keynote Address: Emerging Trends in Software Metrics (WeTSOM’14)

What Metrics Matter?
(And the answer may surprise you)

tim.menzies@gmail.com

Coming soon to an Amazon near you

Sharing Data and Models in Software Engineering
Tim Menzies, Ekrem Kocaguneli, Burak Turhan, Leandro Minku, Fayola Peters

Late 2014

The Art and Science of Analyzing Software Data
Christian Bird, Tim Menzies, Thomas Zimmermann

Late 2015
This talk is in two parts

Part1: a little history
(my unhappiness with past “results”)

Part2: a new view
Data about software projects is not stored in metric1, metric2,...

- But is shared between them in some shared, underlying, shape.

- Not every project has the same underlying simple shape
  - Many projects have different, albeit simple, shapes

- We can exploit that shape, to great effect:
  - For better local predictions
  - For transferring lessons learned
  - For privacy-preserving data mining
So, what metrics to collect?

- Whatever you can get, quickly, cheaply:
  - Then model within the reduced dimensions
  - Then cycle back to the users, for sanity, for clarity, for questions for the next round of analysis

The feedback loops in inductive engineering are about maximizing the intersection between the circles.
PART1:
YE OLDE “RESULTS”
(FROM THE 1990S)
Along time ago...
In a century far, far away...

• We thought the “right name” was inherently power
  – Stranger in a Strange Land (Heinlien)
  – Wizard of Earthsea (LeGuin)
  – Snow Crash (Stephenson)

• Sapir-Whorf hypothesis:
  – The right words let you think better

• And we need such power
  – to avoid the lies and illusions of a cruel and confusing world.
Shotgun correlations


Table 5

<table>
<thead>
<tr>
<th>variables</th>
<th>6</th>
<th>8</th>
<th>10</th>
<th>12</th>
<th>14</th>
<th>16</th>
<th>18</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0.7069</td>
<td>0.6042</td>
<td>0.5377</td>
<td>0.4917</td>
<td>0.4550</td>
<td>0.4239</td>
<td>0.3905</td>
<td>0.3756</td>
</tr>
<tr>
<td>5</td>
<td>0.7721</td>
<td>0.6804</td>
<td>0.6022</td>
<td>0.5493</td>
<td>0.5080</td>
<td>0.4757</td>
<td>0.4508</td>
<td>0.4274</td>
</tr>
<tr>
<td>6</td>
<td>0.8118</td>
<td>0.7184</td>
<td>0.6447</td>
<td>0.5834</td>
<td>0.5429</td>
<td>0.5069</td>
<td>0.4844</td>
<td>0.4621</td>
</tr>
<tr>
<td>7</td>
<td>0.8467</td>
<td>0.7466</td>
<td>0.6807</td>
<td>0.6206</td>
<td>0.5734</td>
<td>0.5424</td>
<td>0.5094</td>
<td>0.4869</td>
</tr>
<tr>
<td>8</td>
<td>0.8673</td>
<td>0.7768</td>
<td>0.6971</td>
<td>0.6505</td>
<td>0.5992</td>
<td>0.5609</td>
<td>0.5327</td>
<td>0.5063</td>
</tr>
<tr>
<td>9</td>
<td>0.8840</td>
<td>0.7947</td>
<td>0.7185</td>
<td>0.6678</td>
<td>0.6134</td>
<td>0.5832</td>
<td>0.5481</td>
<td>0.5223</td>
</tr>
<tr>
<td>10</td>
<td>0.8967</td>
<td>0.8065</td>
<td>0.7381</td>
<td>0.6821</td>
<td>0.6360</td>
<td>0.5970</td>
<td>0.5666</td>
<td>0.5382</td>
</tr>
<tr>
<td>11</td>
<td>0.9052</td>
<td>0.8213</td>
<td>0.7570</td>
<td>0.6958</td>
<td>0.6493</td>
<td>0.6100</td>
<td>0.5801</td>
<td>0.5510</td>
</tr>
<tr>
<td>12</td>
<td>0.9114</td>
<td>0.8376</td>
<td>0.7645</td>
<td>0.7137</td>
<td>0.6614</td>
<td>0.6261</td>
<td>0.5908</td>
<td>0.5610</td>
</tr>
<tr>
<td>13</td>
<td>0.9214</td>
<td>0.8443</td>
<td>0.7800</td>
<td>0.7233</td>
<td>0.6743</td>
<td>0.6374</td>
<td>0.6050</td>
<td>0.5731</td>
</tr>
<tr>
<td>14</td>
<td>0.9267</td>
<td>0.8544</td>
<td>0.7840</td>
<td>0.7306</td>
<td>0.6813</td>
<td>0.6388</td>
<td>0.6110</td>
<td>0.5888</td>
</tr>
<tr>
<td>15</td>
<td>0.9343</td>
<td>0.8624</td>
<td>0.7654</td>
<td>0.7362</td>
<td>0.6929</td>
<td>0.6549</td>
<td>0.6190</td>
<td>0.5631</td>
</tr>
</tbody>
</table>

The table shows the average maximum correlation r for all random variables across all combinations for different sample sizes.
We believe that software metrics should be treated as part of an engineering discipline—metrics should be evaluated (validated) to determine whether they measure what they purport to measure prior to using them. Furthermore, if metrics are to be of greatest utility, the validation should be performed in terms of the quality functions (quality assessment, control, and prediction) that the metrics are to support.

Section V some comments are made about future research directions.
The 1990’s obsession: What metrics to collect?

<table>
<thead>
<tr>
<th>Size</th>
<th>SLOC</th>
<th>% Code Modified</th>
<th>% Code Modified</th>
<th>% Integration Required</th>
<th>Assessment and Anomalies (0% - 8%)</th>
<th>Software Understanding (6% - 50%)</th>
<th>Unfamiliarity (0.1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>New</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Revis</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Modif</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Rate each cost driver below from Very Low (VL) to Extra High (EH). For HELP on each cost driver, see name:

**Scale Drivers**
- **Precededness**
  - Required Reliability: VL, L, N, H, VH
  - Database Size: VL, L, N, H, VH
- **Development Flexibility**
  - Product Complexity: VL, L, N, H, VH
- **Architecture/Risk Resolution**
  - Requisite Reuse: VL, L, N, H, VH
- **Team Cohesion**
  - Documentation: VL, L, N, H, VH
- **Process Maturity**

**Product Attributes**
- **Required Reliability**
  - Execution Time Constraint: N, H, VH, EH
  - Main Storage Constraint: N, H, VH, EH
  - Platform Volatility: L, N, H, VH
- **Product Complexity**
  - Analyst Capability: VL, L, N, H, VH
  - Programmer Capability: VL, L, N, H, VH
  - Personnel Continuity: VL, L, N, H, VH
  - Application Experience: VL, L, N, H, VH
  - Platform Experience: VL, L, N, H, VH
- **Language and Toolset Experience**
  - Use of Software Tools: VL, L, N, H, VH
  - Multisite Development: VL, L, N, H, VH
  - Required Development Schedule: VL, L, N, H, VH
How to Design a Metrics Repository (mid-1990s)

• RA-1: process-centric [1]
• TAME resource model: resource-centric [2]
• Harrison model: product centric [3]


Battle lines were drawn

**Blood was split**

- It is shown that a collection of nine properties suggested by E.J. Weyuker is inadequate for determining the quality of a software complexity measure.

- A complexity measure which satisfies all nine of the properties, but which has absolutely no practical utility in measuring the complexity of a program is presented.

- It is concluded that satisfying all of the nine properties is a necessary, but not sufficient, condition for a good complexity measure.

**And again**

- Measurement theory is used to highlight both weaknesses and strengths of software metrics work, including work on metrics validation.

- We identify a problem with the well-known Weyuker properties, but also show that a criticism of these properties by Cherniavsky and Smith is invalid.

- We show that the search for general software complexity measures is doomed to failure.


Software measurement: A Necessary Scientific Basis, Norman Fenton, IEEE TSE 30(3), 1994
And the eventual winner?

• No one

• When the dust settled, no one really cared.

• Norman Fenton abandoned, renounced, his prior work on metrics.

• The IEEE Metrics conference got cancelled
  – subsumed by EMSE
  – R.I.P.
PART2:
A NEW VIEW
Looking in the wrong direction?

- SE project data = surface features of an underlying effect
- Stop fussing on surface details (mere metrics)
- Go beneath the surface
Reflect **LESS** on raw dimensions

(a) 3-D data  (b) 2-D data  (c) Projected space
Reflect **MORE** on **INTRINSIC** dimensions

- Levina and Bickel report that it is possible to simplify seemingly complex data:
  
  “... the only reason any methods work in very high dimensions is that, in fact, the data are not truly high-dimensional. Rather, they are embedded in a high-dimensional space, but can be efficiently summarized in a space of a much lower dimension ...”

If SE data compresses in this way then….

• In compressed space, many measures tightly associated
  – Does not matter exactly what you collect
  – Since they will map the same structures

• So collect what you can:
  – As fast as you can
  – Then model within the reduced dimensions

• The “7M” Hypothesis:
  – Menzies mentions that many measures mean much the same thing
How to test for 7M

• Specifics do not matter
• But general shape does
• Examples:
  – Instability in “what matters most”
  – Intrinsic dimensionality
  – Active learning
  – Transfer learning
  – Filtering and wrapping
  – Privacy algorithms
How to test for 7M

• Specifics do not matter
• But general shape does
• Examples:
  – Instability in “what matters most”
  – Intrinsic dimensionality
  – Transfer learning
  – Filtering and wrapping
  – Privacy algorithms
  – Active learning
Raw dimensions problematic (for effort estimation)

• Conclusion instability
• Learning effort = \( b_0 + b_1 \times x_1 + b_2 \times x_3 + \)
• 20 times * 66% of the data
  – Record the learned “b” values
• NASA93 (effort data)
### Raw dimensions problematic (for defect prediction)

<table>
<thead>
<tr>
<th>ref</th>
<th>cbo</th>
<th>rfc</th>
<th>lcom</th>
<th>dit</th>
<th>noc</th>
<th>wmc</th>
<th>#projects</th>
<th>size</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6</td>
<td>95-201 classes</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>12</td>
<td>86 classes (3-12kloc)</td>
<td>commercial telecom</td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1700 classes (110kloc)</td>
<td>student</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8</td>
<td>113 classes</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>114 classes</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>83 classes</td>
<td>commercial: lalo (c++)</td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>32 classes</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>42-69 classes</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>5 classes</td>
<td>commercial java word proc.</td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>92 classes</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>123 classes (34kloc)</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>706 classes</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>145 classes</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>3677 classes</td>
<td>open source:mozilla</td>
</tr>
<tr>
<td>27</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>113 classes</td>
<td>student</td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>64 classes</td>
<td>sales and cd-selection system</td>
</tr>
<tr>
<td>31</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>3344 modules (2mloc)</td>
<td>commercial telecom c++</td>
</tr>
<tr>
<td>32</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>395 classes</td>
<td>commercial telecom c++</td>
</tr>
<tr>
<td>33</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1412 classes</td>
<td>open source:mozilla</td>
</tr>
<tr>
<td>34</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>9713 classes</td>
<td>eclipse 2.0, 2.1</td>
</tr>
<tr>
<td>35</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>74 classes</td>
<td>kcl-nasa</td>
</tr>
<tr>
<td>36</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>145 classes</td>
<td>commercial java xml editor</td>
</tr>
<tr>
<td>37</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>kcl-nasa</td>
<td>commercial telecom c++</td>
</tr>
<tr>
<td>38</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>145 classes</td>
<td>student</td>
</tr>
<tr>
<td>39</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>294 classes</td>
<td>kcl-nasa</td>
</tr>
<tr>
<td>40</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>total</td>
<td>18</td>
<td>20</td>
<td>11</td>
<td>11</td>
<td>8</td>
<td>17</td>
<td>6</td>
<td>95-201 classes</td>
<td></td>
</tr>
</tbody>
</table>

**Total percents:** **"*" denotes majority conclusion in each column.**

<table>
<thead>
<tr>
<th>+</th>
<th>* 64%</th>
<th>* 71%</th>
<th>* 39%</th>
<th>39%</th>
<th>29%</th>
<th>* 61%</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>14%</td>
<td>11%</td>
<td>25%</td>
<td>* 50%</td>
<td>* 57%</td>
<td>14%</td>
</tr>
</tbody>
</table>

**KEY:**
- **Strong consensus (over 2/3rds)**
- **Some consensus (less than 2/3rds)**
- **Weak consensus (about half)**
- **No consensus**

Fig. 2. Contradictory conclusions from OO-metrics studies for defect prediction. Studies report significant ("+") or irrelevant ("-" ) metrics verified by univariate prediction models. Blank entries indicate that the corresponding metric is not evaluated in that particular study. Colors comment on the most frequent conclusion of each column. CBO= coupling between objects; RFC= response for class (#methods executed by arriving messages); LCOM= lack of cohesion (pairs of methods referencing one instance variable, different definitions of LCOM are aggregated); NOC= number of children (immediate subclasses); WMC= #methods per class.

By the way, same instabilities for social metrics

**Results from Helix repo**

- Many classes are “write once”
  - A few OO classes are “rewrite many”

- Defect defectors that take into account this developer social interaction
  - perform very well indeed
  - Near optimum

**Results for AT&T**

- Studied patterns of programmer interaction with the code

- Not a major influence on defects

Lumpe, Vasa, Menzies, Rush, Turhan, Learning better inspection optimization policies International Journal of Software Engineering and Knowledge Engineering 22(05), 621-644, 2012

Weyuker, Ostrand, Bell. 2008. Do too many cooks spoil the broth? Using the number of developers to enhance defect prediction models. Empirical Softw. Eng. 13, 5 (October 2008), 539-559
How to test for 7M

• Specifics do not matter
• But general shape does
• Examples:
  – Instability in “what matters most”
  – **Intrinsic dimensionality**
  – Active learning
  – Transfer learning
  – Filtering and wrapping
  – Privacy algorithms
Intrinsic dimensionality (in theory)

\[ C(r) = \frac{2}{n(n-1)} \sum_{i=1}^{n} \sum_{j=i+1}^{n} \begin{cases} 
1, & \text{if } \text{dist}(x_i, x_j) < r. \\
0, & \text{otherwise.}
\end{cases} \]


Figure 1: The intrinsic dimensionality of many SE defect data sets from large OO Java systems is usually less than two.
Intrinsic dimensionality (in practice)

- Defect projects, open source JAVA projects

- **TRAIN:**
  - Project 21 features onto two synthesized
    - using FASTMAP
    - X= First PCA component
    - Y= right angles to X
  - Recursively divide two dimensions (at median)
    - Stopping a SQRT(N)
  - In each grid, replace N projects with median centroid

- **TEST:**
  - Estimate = interpolate between 2 near centroids

- For 10 data sets, 5*5 cross-val
  - Performs no worse, and sometimes better, than Random forests, NaiveBayes

- **Conclusion:**
  - 21 dimensions can map to two without loss of signal

How to test for 7M

• Specifics do not matter
• But general shape does
• Examples:
  – Instability in “what matters most”
  – Intrinsic dimensionality
  – **Active learning**
  – Transfer learning
  – Filtering and wrapping
  – Privacy algorithms
Active learning in effort estimation

- If difficult to find actual project effort, ask that for fewest projects
- Put aside a hold-out set
- Prune columns that are most often other column’s nearest neighbors
- Sort rows by how often they are other people’s nearest neighbor
- For first “I” rows in sort
  - Train (then test on hold out)
  - Stop when no performance gain after N new rows

How to test for 7M

• Specifics do not matter
• But general shape does
• Examples:
  – Instability in “what matters most”
  – Intrinsic dimensionality
  – Active learning
  – Transfer learning
  – Filtering and wrapping
  – Privacy algorithms
Between Turkish Toasters AND NASA Space Ships
Q: How to TRANSFER Lessons Learned?

- Ignore most of the data
- relevancy filtering: *Turhan ESEj’09; Peters TSE’13*
- variance filtering: *Kocaguıneli TSE’12, TSE’13*
- performance similarities: *He ESEM’13*

- Contort the data
- spectral learning (working in PCA space or some other rotation) *Menzies, TSE’13; Nam, ICSE’13*

- Build a **bickering committee**
- Ensembles *Minku, PROMISE’12*
BTW, Sometimes, TRANSFER better than LOCAL

Minku: PROMISE’12

Nam: ICSE’13

<table>
<thead>
<tr>
<th>Project</th>
<th>#systems</th>
<th>Mean F-measures</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Within</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>0.53</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>0.42</td>
</tr>
</tbody>
</table>

Peters: TSE’13

![Graphs showing g-measure vs IPR for skarbonka, tomcat, and velocity-1.4 projects.]
How to test for 7M

• Specifics do not matter
• But general shape does
• Examples:
  – Instability in “what matters most”
  – Intrinsic dimensionality
  – Active learning
  – Transfer learning
  – Filtering and wrapping
  – Privacy algorithms
Some technology: feature selectors

- **Wrappers**
  - Slow: $O(2^N)$ for $N$ features
  - Selection biased by target learner

- **Filters**
  - Faster (some even linear time)
  - Selection may confuse target

M.A. Hall and G. Holmes, Benchmarking Attribute Selection Techniques for Discrete Class Data Mining, IEEE Transactions On Knowledge And Data Engineering 15(6) 1437-1447. 2--3
Filter results

- Data from Norman Fenton’s Bayes nets discussing software defects = yes, no

- Give classes $x, y$
  - $F_x, F_y$
    - frequency of discretized ranges in $x, y$
  - Log Odds Ratio [1]
    - $\log(F_x/F_y)$
    - Is zero if no difference in $x, y$

- Most variables do not contribute to determination of defects

---

Filter results (more)

- Defect prediction, NASA data
- Baseline: learn defect model on all data (McCabes + Halstead + LOC measures)
- Filter = sort columns on “info gain”
- Experiment = use the first N items in the sort, stopping when recall, false alarm, same as baseline

Filter results on defect data: many different features equally as good for prediction

- Consistent with this data “pinging” a much lower dimensional space

Fig. 14. InfoGain for KC3 attributes. Calculated from (2). Lines show means and t-bars show standard deviations after 10 trials on 90 percent of the training data (randomly selected).
Some Wrapper results: effort estimation, NASA data

- $X = f(a,b,c,\ldots)$
- $X$’s variance comes from $a,b,c$
- If less $a,b,c$
  - then less confusion about $X$
- E.g. effort estimation
- $\text{Pred}(30) = \%$estimates within 30% of actual

How to test for 7M

• Specifics do not matter
• But general shape does
• Examples:
  – Instability in “what matters most”
  – Intrinsic dimensionality
  – Active learning
  – Transfer learning
  – Filtering and wrapping
  – Privacy algorithms
Peter’s Power Principle
(for row and column pruning)

Filtering via range “power”

- Divide data with N rows into
  - one region for classes x,y, etc
- For each region x, of size nx
  - px = nx/N
  - py (of everything else) = (N-nx)/N
- Let Fx and Fy be frequency of range r in
  (1) region x and (2) everywhere else
- Do the Bayesian thing:
  - a = Fx * px
  - b = Fy * py
- Power of range r for predicting x is:
  - POW[r,x] = a²/(a+b)

Pruning

- Column pruning
  - Sort columns by power of column (POC)
    - POC = max POW value in that column
- Row pruning
  - Sort rows by power of row (POR)
  - If row is classified as x
    - POR = Prod( POW[r,x] for r in row )
- Keep 20% most powerful rows and columns:
  - 0.2 * 0.2 = 0.04
  - i.e. 4% of the original data
Q: What does that look like?
A: Empty out the “billiard table”

• This is a privacy algorithm:
  – CLIFF: prune X% of rows, we are 100-X% private
  – MORPH: mutate the survivors no more than half the distance to their nearest unlike neighbor
  – One of the few known privacy algorithms that does not damage data mining efficacy

Advanced privacy methods

Incremental learning

• Pass around the reduced data set
• “Alien”: new data is too “far away” from the reduced data
  – “Too far”: 10% of separation most distance pair
• If anomalous, add to cache
  – For defect data, cache does not grow beyond 3% of total data

Privacy-preserving Cross-company Learning

• LACE : Learn from N software projects
  – Mixtures of open+closed source projects
• As you learn, play “pass the parcel”
  – The cache of reduced data
• Each company only adds its “aliens” to the passed cache
  – Morphing as they goes
• Each company has full control of privacy
• Generated very good defect predictors

ASE’14: submitted

SUMMARY & CONCLUSIONS
Underlying dimensions more interesting than raw dimensions

• We can ignore most of the data
  – And still find the signal
  – Filters, active learning
  – Sometimes even enhancing the signal
    • Wrappers

• We can significantly and usefully contort the data
  – And still get our signal
  – E.g. transfer learning, privacy
Data about software projects is not stored in metric1, metric2,...

• But is shared between them in some shared, underlying, shape.

• Not every project has the same underlying simple shape
  – Many projects have different, albeit simple, shapes

• We can exploit that shape, to great effect:
  – For better local predictions
  – For transferring lessons learned
  – For privacy-preserving data mining
We were looking in the wrong direction

- SE project data = surface features of an underlying effect
- Stop fussing on surface details (mere metrics)
- Go beneath the surface
Reflect **LESS** on raw dimensions

(a) 3-D data  (b) 2-D data  (c) Projected space
Reflect MORE on INTRINSIC dimensions

- Levina and Bickel report that it is possible to simplify seemingly complex data:
  - “... the only reason any methods work in very high dimensions is that, in fact, the data are not truly high-dimensional. Rather, they are embedded in a high-dimensional space, but can be efficiently summarized in a space of a much lower dimension ...”

If SE data compresses in this way then....

• The “7M” Hypothesis:
  – Menzies mentions that many measures mean much the same thing

• In compressed space, many measures tightly associated
  – Does not matter exactly what you collect
  – Since they will map the same structures
So, what metrics to collect?

• Whatever you can get, quickly, cheaply:
  – Then model within the reduced dimensions
  – Then cycle back to the users, for sanity, for clarity, for questions for the next round of analysis

The feedback loops in inductive engineering are about maximizing the intersection between the circles.
THE END
The End

No, it isn’t
Speculation

SHOULD WE STUDY L-SYSTEM?
Bracketed L-systems

- In drawing branching structures using the turtle interpreter it is necessary to reposition the turtle at the base of a branch after the drawing of the branch itself.
- Bracketed L-systems facilitate this task.
- Two new symbols are defined:
  - [ Save current state of the turtle (position, orientation, color, thickness, etc.).
  - ] Restore the state of the turtle using the last saved state (no line is drawn).
- Bracketed L-systems are also useful to define hierarchical networks using the graph interpreter.

\[ \delta = 29^\circ, \quad A = \{ F, +, -, [ , ] \} \]
\[ \omega = F \]
\[ p = F \rightarrow F [ + F ] F [- F ] [ + F ] [ - F ] ] F \]

Stochastic L-systems

• All plants generated by the same L-system are identical, but in nature individuals are not identical.
• Specimen-to-specimen variation can be modeled by introducing production probabilities. For every symbol, there is one or more production rules with an associated probability. The sum of all probabilities over the same symbol must be 1

\[ \delta = 29^\circ, \ A = \{ F, +, -, [ ], ] \} \]
\[ \omega = F \]
\[ p_1 = F^{1/3} \quad F[+F]F[-F]F \]
\[ p_2 = F^{1/3} \quad F[-F]F[+F]F \]
\[ p_3 = F^{1/3} \quad F[-FF-F]F \]

Companion slides for the book *Bio-Inspired Artificial Intelligence: Theories, Methods, and Technologies* by Dario Floreano and Claudio Mattiussi, MIT Press
Applications to computer graphics

Companion slides for the book *Bio-Inspired Artificial Intelligence: Theories, Methods, and Technologies* by Dario Floreano and Claudio Mattiussi, MIT Press
Is this really the end?
End of my tale