Measure Software – and its Evolution – Using Information Content

Tom Arbuckle
tom.arbuckle@ieee.org

Electronic and Computer Engineering
University of Limerick
IRELAND

IWPSE-EVOL: 25/Aug/09
Software Releases
Measuring Software Evolution

Software Releases

Measurement

"Metrics" OR mathematical distance measures.
Measuring Software Evolution

Software Releases

Software Releases

Measurement

⇒ “Metrics” OR mathematical distance measures.
Questions:

- **What** are we measuring?
- **How** do we measure it?

For this talk: mathematical metric = metric ≠ software metric
Q1. What is **software**?
Q1. What is software?

A. “Information encoding design decisions.”
Q1. What is software?

A. “Information encoding design decisions.”

SO …

Q2. How do we measure information?
Q1. What is software?

A. “Information encoding design decisions.”

SO …

Q2. How do we measure information?

A.1. Shannon?
Q1. What is *software*?

A. “*Information* encoding design decisions.”

SO …

Q2. How do we measure *information*?

A.1. *Shannon*? NOT QUITE (in this case).
Q1. What is **software**?

A. “**Information** encoding design decisions.”

SO …

Q2. How do we measure **information**?


A.2. **Kolmogorov**?
What is software?

- Q1. What is **software**?
  - A. “**Information** encoding design decisions.”
    - SO …

- Q2. How do we measure **information**?
But, but, but, . . .

Q. What about: LOC, DIT, NOM, ... ⟨useful, practical, software metrics⟩?
But, but, but, . . .

- Q. What about: LOC, DIT, NOM, ... \langle useful, practical, software metrics\rangle?
- Q. Chidamber & Kemerer? Halstead? Function Points?
Q. What about: LOC, DIT, NOM, ... ⟨useful, practical, software metrics⟩?
Q. Chidamber & Kemerer? Halstead? Function Points?
Q. Throw our software metrics away?
But, but, but, . . .

- Q. What about: LOC, DIT, NOM, ... ⟨useful, practical, software metrics⟩?
- Q. Chidamber & Kemerer? Halstead? Function Points?
- Q. Throw our software metrics away?

- A. When we want to “know” if it will rain tomorrow, we can . . .
But, but, but, . . .

Q. What about: LOC, DIT, NOM, ... ⟨useful, practical, software metrics⟩?
Q. Chidamber & Kemerer? Halstead? Function Points?
Q. Throw our software metrics away?

A. When we want to “know” if it will rain tomorrow, we can . . .
   . . . look at the clouds . . .
But, but, but, . . .

Q. What about: LOC, DIT, NOM, ... ⟨useful, practical, software metrics⟩?
Q. Chidamber & Kemerer? Halstead? Function Points?
Q. Throw our software metrics away?

A. When we want to “know” if it will rain tomorrow, we can . . .
   - . . . look at the clouds . . .
   - . . . (use our knowledge of physics and meteorology and) look at our barometer.
Q. What about: LOC, DIT, NOM, ... 〈useful, practical, software metrics〉?
Q. Chidamber & Kemerer? Halstead? Function Points?
Q. Throw our software metrics away?

A. We have things that work but we don’t know why.
But, but, but, . . .

- Q. What about: LOC, DIT, NOM, ... ⟨useful, practical, software metrics⟩?
- Q. Chidamber & Kemerer? Halstead? Function Points?
- Q. Throw our software metrics away?

- A. We have things that work but we don’t know why.
- If we knew why, . . .
Q. What about: LOC, DIT, NOM, ... ⟨useful, practical, software metrics⟩?
Q. Chidamber & Kemerer? Halstead? Function Points?
Q. Throw our software metrics away?

A. We have things that work but we don’t know why.
If we knew why, . . .
   . . . we might not need them.
Q. What about: LOC, DIT, NOM, ... ⟨useful, practical, software metrics⟩?
Q. Chidamber & Kemerer? Halstead? Function Points?
Q. Throw our software metrics away?

A. We have things that work but we don’t know why.
If we knew why, . . .
   . . . we might not need them.
   . . . we could in principle unify/compare them all.
Q. What about: LOC, DIT, NOM, ... ⟨useful, practical, software metrics⟩?
Q. Chidamber & Kemerer? Halstead? Function Points?
Q. Throw our software metrics away?

A. We have things that work but we don’t know why.
   If we knew why, . . .
      . . . we might not need them.
      . . . we could in principle unify/compare them all.
      . . . we could move from description/explanation \(\rightarrow\) prediction
Q. Why Kolmogorov/Shannon?
Kolmogorov complexity, Shannon entropy

Q. Why Kolmogorov/Shannon?

A1. **Shannon**: significance of symbols from known alphabet.

Symbols? Encoding? Probability distributions?
Kolmogorov complexity, Shannon entropy

Q. Why Kolmogorov/Shannon?

A1. **Shannon**: significance of symbols from known alphabet.

A2. **Kolmogorov**: information content of objects themselves.

**Definition**

Kolmogorov complexity: **length(size)** of shortest binary program to produce “**object**” as output when run on universal computer such as a Turing machine.
Q. Why Kolmogorov/Shannon?

A1. **Shannon**: significance of symbols from known alphabet.

A2. **Kolmogorov**: information content of objects themselves.

Example
Source probabilistically emits two messages.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>
Why relative information content?
Why relative information content?

A 1 million bits
...10100011001010...

B 1 million bits
...10100011001010...
Why relative information content?

\[ N_{\text{ID}}(x, y) = \max \{ K(x | y), K(y | x) \} \]

\[ \max \{ K(x), K(y) \} \]

Normalised Information Distance:
number of bits of shared information per bit of information of the larger string.
Why relative information content?

A 1 million bits
...10100011001010...

A’ 1 thousand bits
...10100011001010...

B 1 million bits
...10100011001010...

B’ 1 thousand bits
...10010100101001...

Relative information content

\[
\text{Relative information content} = \frac{\text{Shared information}}{\text{Information of larger string}}
\]

\[
\text{NID}((x, y)) = \max\{K(x|y), K(y|x)\}
\]

\[
\text{Normalised Information Distance: number of bits of shared information per bit of information of the larger string.}
\]
Relative information content

Why relative information content?

\[ NID(x, y) = \frac{\max\{K(x|y), K(y|x)\}}{\max\{K(x), K(y)\}} \]  

Normalised Information Distance: number of bits of shared information per bit of information of the larger string.
Kolmogorov complexity is **uncomputable**: solution?
Uncomputability!? 

- Kolmogorov complexity is uncomputable: solution?
- Approximate Kolmogorov using Shannon OR . . .
Uncomputability!?

- Kolmogorov complexity is uncomputable: solution?
- Approximate Kolmogorov using Shannon OR . . .
- Approximate Kolmogorov using compression . . .
Uncomputability!? 

- Kolmogorov complexity is uncomputable: solution?
- Approximate Kolmogorov using Shannon OR ... 
- Approximate Kolmogorov using compression ... 

\[ NID \rightarrow \]
Kolmogorov complexity is uncomputable: solution?

Approximate Kolmogorov using Shannon OR . . .

Approximate Kolmogorov using compression . . .

\[ NID \rightarrow \]

\[ NCD(x, y) = \frac{C(xy) - \min\{C(x), C(y)\}}{\max\{C(x), C(y)\}}. \] (2)
Uncomputability!?

- Kolmogorov complexity is uncomputable: solution?
- Approximate Kolmogorov using Shannon OR . . .
- Approximate Kolmogorov using compression . . .

\[ NID \rightarrow \]

\[ NCD(x, y) = \frac{C(xy) - \min\{C(x), C(y)\}}{\max\{C(x), C(y)\}}. \quad (2) \]

\( C(x) \): compressed version of \( x \). N.B.: conditions on \( C \).
Uncomputability!?

- Kolmogorov complexity is uncomputable: solution?
- Approximate Kolmogorov using Shannon OR . . .
- Approximate Kolmogorov using compression . . .

\[ \text{NID} \rightarrow \]

\[ NCD(x, y) = \frac{C(xy) - \min\{C(x), C(y)\}}{\max\{C(x), C(y)\}}. \]  

(2)

\(C(x)\): compressed version of \(x\). N.B.: conditions on \(C\).

\(\rightarrow\) proof-of-concept example.
Choose a “representative” artefact to measure.
Choose a “representative” artefact to measure.

- Source code
Choose a “representative” artefact to measure.
- Source code
- Binaries

udev I — Procedure
Choose a “representative” artefact to measure.
- Source code
- Binaries
- Representations of structure, e.g. graphs
Choose a “representative” artefact to measure.
- Source code
- Binaries
- Representations of structure, e.g. graphs
- Representations of behaviour, e.g. execution traces
Choose a “representative” artefact to measure.

- Source code
- Binaries
- Representations of structure, e.g. graphs
- Representations of behaviour, e.g. execution traces
- Others, both static and dynamic . . .
Choose a “representative” artefact to measure.

- Source code
- Binaries
- Representations of structure, e.g. graphs
- Representations of behaviour, e.g. execution traces
- Others, both static and dynamic . . .

Pairwise compare information content.
Choose a “representative” artefact to measure.
- Source code
- Binaries
- Representations of structure, e.g. graphs
- Representations of behaviour, e.g. execution traces
- Others, both static and dynamic...

Pairwise compare information content.

Plot distance matrix.
Choose a “representative” artefact to measure.
- Source code
- Binaries
- Representations of structure, e.g. graphs
- Representations of behaviour, e.g. execution traces
- Others, both static and dynamic...

Pairwise compare information content.

Plot distance matrix.

Look for discontinuities, anomalies
Choose a “representative” artefact to measure.
- Source code
- Binaries
- Representations of structure, e.g. graphs
- Representations of behaviour, e.g. execution traces
- Others, both static and dynamic...

Pairwise compare information content.

Plot distance matrix.

Look for discontinuities, anomalies

Choice: udev program suite, analyze for 141 releases.
Choose a “representative” artefact to measure.
- Source code
- Binaries
- Representations of structure, e.g. graphs
- Representations of behaviour, e.g. execution traces
- Others, both static and dynamic ...

Pairwise compare information content.

Plot distance matrix.

Look for discontinuities, anomalies

Choice: udev program suite, analyze for 141 releases.

Artefact: concatenation of all source (.c, .h) files.
udev II — Results
**udev II — Results**

**Table:** Reasons for major changes

<table>
<thead>
<tr>
<th>Index</th>
<th>Release</th>
<th>Notable Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>6</td>
<td>SCCS files kept in source</td>
</tr>
<tr>
<td>16</td>
<td>16</td>
<td>Removal of SCCS files</td>
</tr>
<tr>
<td>43,44</td>
<td>44,45</td>
<td>No code changes</td>
</tr>
<tr>
<td>53</td>
<td>54</td>
<td>Update klibc with zlib</td>
</tr>
<tr>
<td>73</td>
<td>74</td>
<td>Remove own copy of klibc</td>
</tr>
<tr>
<td>79</td>
<td>80</td>
<td>Replace libsysfs</td>
</tr>
<tr>
<td>98,99</td>
<td>99,100</td>
<td>Almost no code changes</td>
</tr>
<tr>
<td>126</td>
<td>127</td>
<td>libudev info library</td>
</tr>
</tbody>
</table>
Summary & Conclusion

- Software: information representing design decisions.

Measure software – and its evolution – using information content.
Summary & Conclusion

- Software: information representing design decisions.
- Measure information: → information theory

Measure software – and its evolution – using information content.
Software: information representing design decisions.

Measure information: \(\rightarrow\) information theory

- Shannon: not discussing messages but objects

Measure software – and its evolution – using information content.
Summary & Conclusion

- Software: information representing design decisions.
- Measure information: → information theory
  - Shannon: not discussing messages but objects
  - Kolmogorov: information in “objects”

Measure software – and its evolution – using information content.
Software: information representing design decisions.

Measure information: → information theory
   - Shannon: not discussing messages but objects
   - Kolmogorov: information in “objects”

Kolmogorov: uncomputable → approximate

Measure software – and its evolution – using information content.
Summary & Conclusion

- Software: information representing design decisions.
- Measure information: → information theory
  - Shannon: not discussing messages but objects
  - Kolmogorov: information in “objects”
- Kolmogorov: uncomputable → approximate
- Approximation methods becoming available now.

Measure software – and its evolution – using information content.
Summary & Conclusion

- Software: information representing design decisions.
- Measure information: \(\rightarrow\) information theory
  - Shannon: not discussing messages but objects
  - Kolmogorov: information in “objects”
- Kolmogorov: uncomputable \(\rightarrow\) approximate
- Approximation methods becoming available now.
  - Use compression.

Measure software – and its evolution – using information content.
Software: information representing design decisions.

Measure information: → information theory
- Shannon: not discussing messages but objects
- Kolmogorov: information in “objects”

Kolmogorov: uncomputable → approximate

Approximation methods becoming available now.
- Use compression.

Proof-of-concept: try looking at udev

Measure software – and its evolution – using information content.
Software: information representing design decisions.

Measure information: → information theory
- Shannon: not discussing messages but objects
- Kolmogorov: information in “objects”

Kolmogorov: uncomputable → approximate

Approximation methods becoming available now.
- Use compression.

Proof-of-concept: try looking at udev
- Can see evolution

Measure software – and its evolution – using information content.
Software: information representing design decisions.

Measure information: \( \rightarrow \) information theory
- Shannon: not discussing messages but objects
- Kolmogorov: information in “objects”

Kolmogorov: uncomputable \( \rightarrow \) approximate

Approximation methods becoming available now.
- Use compression.

Proof-of-concept: try looking at udev
- Can see evolution
- Could also use for behaviour, structure, others . . .

Measure software – and its evolution – using information content.
Is the case study I chose representative?
Feedback questions

1. Is the case study I chose representative?
2. What other case studies do you propose for my work?
Feedback questions

1. Is the case study I chose representative?
2. What other case studies do you propose for my work?

Thank you.

Tom Arbuckle

tom.arbuckle@ieee.org
http://www.ece.ul.ie/homepage/Tom_Arbuckle/


Relation between Entropy and Kolmogorov Complexity . . .

- Expected Kolmogorov Complexity ≈ Shannon entropy

\[ 0 \leq \left( \sum_x f(x)K(x) - H(x) \right) \leq K(f) + O(1) \]  

where \( f(x) = P(X = x) \) on \( \chi = \{0, 1\}^* \)
and \( H(X) = -\sum_x f(x) \log f(x) \)

- Expected algorithmic mutual information ≈ probabilistic mutual information
udev — Result Comparison

udev versions up to 142 [ppmd]
Measure Software – and its Evolution – Using Information Content

Tom Arbuckle

Introduction
Exposition
Proof-of-Concept
Example
Feedback
Questions
Extras

udev — Result Comparison
udev — Result Comparison