Big data. Little Brain.
Little data. Big brain.
(with apologies to Dr. Seuss)

Gail C. Murphy
University of British Columbia
Tasktop Technologies
big data* >> little data

*big data refers to much much more data than humans can consume
fact #1: software engineering is about solving human problems
fact #2: software developments generate big data over time
fact #3: human cognition has limits
premise:
our “solutions” often exceed human cognition

<cartoon removed due to license restrictions>
big data. little brain.
big brain. little data.

duplicate bug examples to derive “formula”
big data $\gg$ little data $\rightarrow$ little brain
stories about formula properties
transformational & situational factors

take away
duplicate bug problem
how does a developer realize/find a new bug is a duplicate?
big data → big brain

1. duplicate bug search with keywords

I like the change to place tabs in the title bar on Windows and would like to see the same thing implemented on macOS. Right now the title bar just displays duplicate information (e.g. the same text as the current tab), I would prefer to use this screen space for something useful. This change would also make Firefox use less vertical screen space than Safari, instead of more, so that would be a bonus too.

This list is too long for Bugzilla's little mind; the Next/Prev/First/Last buttons won't appear on individual pages unless...

<table>
<thead>
<tr>
<th>Product</th>
<th>Component</th>
<th>Status</th>
<th>Resolution</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firefox</td>
<td>OS</td>
<td>NEW</td>
<td>---</td>
<td>mozila does not support Operating System aspects in Bugzilla.</td>
</tr>
<tr>
<td>Mozilla</td>
<td>Bugzilla</td>
<td>NEW</td>
<td>---</td>
<td>Bugzilla does not support Mozilla in Bugzilla.</td>
</tr>
<tr>
<td>Chrome</td>
<td>Bar</td>
<td>OPEN</td>
<td>---</td>
<td>Chrome does not support Bar in Bugzilla.</td>
</tr>
<tr>
<td>Safari</td>
<td>UI</td>
<td>OPEN</td>
<td>---</td>
<td>Safari does not support UI in Bugzilla.</td>
</tr>
</tbody>
</table>

**bug #56162:**
- Reported in 2000
- 16 comments
- 2x as long as #136422

**bug #136422:**
- Reported in 2000
- 32 comments
- 2x as long as #56162
big data >> little data → big brain

2. duplicate bug search with natural language recommender

36-50% recall

[Runenson et. Al, ICSE 2007]
[Hiew, MSc Thesis, 2006]
big data >> little data → big brain

3. duplicate bug search with natural language + execution recommender

67-93% recall

[Wang et. Al, ICSE 2008]
big data >> little data → little brain

4. duplicate bug search with recommender & summarizer

Summary:
View source is broken.

The second time I use it, the View source window fails to open, but I get an hourglass-and-pointer cursor.

I can’t reproduce comment 0 with Mozilla/5.0

[Rastkar & Murphy, ICSE 2010]
interludes

big data >> little data
  significant reduction in data
  a human considers

big brain  versus little brain
  amount of cognitive and physical
  activity required to perform task
big data $\gg$ little data $\rightarrow$ little brain
big data. little brain.
big brain. little data.

duplicate bug examples to “derive formula”
big data $\gg$ little data $\rightarrow$ little brain
stories about formula properties
transformation & situational factors

take away
stories

software reflexion model
hipikat
mylyn
spyglass

to get at formula properties
story #1: software reflexion model
three case studies,
including one at Microsoft

[Murphy, Notkin, Sullivan, FSE 1995]
high-level model of part of Microsoft Excel

[Murphy & Notkin, Computer 1997]
example (not Excel) low-level model
(Excel: 15,000 functions/77,746 calls)

Diagram from http://rixstep.com/1/1/20070206,00.shtml
[ file = ^shtreal\.c function=foo mapTo=Graph ]
[ file = ^shtreal\.c mapTo=Sheet ]
[ file = ^textfl1[ez]\c$ mapTo=File ]

mapping

blue entry is illustrative
reflexion model (partial)
big data >> little data → little brain

big data = low-level model
little data = high-level model

transformational properties
  low-cost (albeit manual)
  assessable completeness
  predictable
high-level model \rightarrow \text{mapping} \rightarrow \text{reflexion model} \rightarrow \text{mapped low-level model} \leftarrow \text{unmapped low-level model}

- low-cost because mapping can be specified simply, incrementally and partially.
- easy to assess completeness.
- predictable.
story #1: software reflexion models

big data >> little data → little brain
low-cost (manual)
assessable completeness
predictable
story #2: hipikat

wizard-of-oz case study
multiple case study (long sessions)
manual precision/recall
Bug 116 - [CVS Repo View] "Version from Stream" should refresh view

Bug#: 116
Version: 2.0
OS/Version: All
Severity: normal
Resolution: FIXED
Target Milestone: 2.1 M4
URL:

Summary:
[CVS Repo View] "Version from Stream" should refresh view (1GENEDO)

Keywords: investigate

Description:
(From Julian J)
When the user selects the "Version from stream..." action in the Repository view, I'd recommend you refresh the view once the action has been completed.

NOTES:
RM (6/1/01 11:26:01 AM)

Cubranic, Murphy, Booth and Singer, TSE 2005]
<table>
<thead>
<tr>
<th>Location in the IDE</th>
<th>Artifact type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bug report open in the Bugzilla editor</td>
<td>change task</td>
</tr>
<tr>
<td>CVS-managed file open in the Java editor</td>
<td>file version</td>
</tr>
<tr>
<td>File in the CVS Repository view</td>
<td>file version</td>
</tr>
<tr>
<td>Revision in the CVS Resource History view</td>
<td>file version</td>
</tr>
<tr>
<td>CVS-managed file in the workspace Navigator</td>
<td>file version</td>
</tr>
<tr>
<td>Item recommended in the Hipikat Results view</td>
<td>item’s type</td>
</tr>
<tr>
<td>Bugzilla search match in the Search results view</td>
<td>change task</td>
</tr>
<tr>
<td>Java class or method in Outline and Hierarchy views</td>
<td>file version</td>
</tr>
</tbody>
</table>

+ text
(e.g., exception trace)
big data >> little data → big brain

big data = project repositories
little data = recommendations

transformational properties
low-cost (automatic)
situational factors
context (automatic)
low-cost because project memory auto-updates

multiple entry points provide auto-context
story #2: hipikat

big data $\gg$ little data $\rightarrow$ big brain

low-cost (automatic)

context (automatic)
story #3: mylyn
three field studies
adoption in practice

[Kersten & Murphy, FSE 2006]
demo
big data >> little data → little brain

big data = workspace information
little data = focused workspace information

transformational properties
- low-cost (automatic)
- assessable completeness
- predictable

situational factors
- pervasive
- fits in workflow
complete, predictable, automatic (with manual override)

pervasive
folding/ & content assist
test
workflow
change sets

private void initContextChangeSets() {
    // For testing.
}

public void clearActiveChangeSets() {
    activeChangeSets.clear();
}

public IResource[] getResourc

public void context

public void context

private ITask get
List<ITask>
if (activeTas
story #3: mylyn

big data $\gg$ little data $\rightarrow$ little brain
low-cost (automatic)
assessable completeness
predictable

pervasive
fits in workflow
story #4: spyglass
longitudinal case study
controlled lab study

[Viriyakattiyaïaporn & Murphy, CASCON 2010]
Map axisStateMap = new HashMap();

// draw the top axes
double minY = dataArea.getMinY();
double minY = dataArea.getMinY() - this.axisOffset.calculateTor...
big data $\gg$ little data $\rightarrow$ big brain

big data = user interactions in workspace
little data = recommended commands

transformational properties
low-cost (automatic)

situational factors
(mostly) fits in workflow
(mostly) fits in workflow
story #4: spyglass

big data >> little data → big brain

low-cost (automatic)

(mostly) fits in workflow
lessons
lessons about transformational factors

big data >> little data

low-cost (manual or automatic)
assessable completeness
predictable
<table>
<thead>
<tr>
<th></th>
<th>reflexion model</th>
<th>hipikat</th>
<th>mylyn</th>
<th>spyglass</th>
</tr>
</thead>
<tbody>
<tr>
<td>low-cost</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>assessable completeness</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>predictable</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>

need sufficiently high precision/recall?
lessons about situational factors

little data $\rightarrow$ little brain

context (automatic)
pervasive
workflow
<table>
<thead>
<tr>
<th>context (automatic)</th>
<th>reflexion model</th>
<th>big brain</th>
<th>spyglass</th>
</tr>
</thead>
<tbody>
<tr>
<td>context</td>
<td>provide intent automatically?</td>
<td>how to handle specialized tools?</td>
<td></td>
</tr>
<tr>
<td>pervasive</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>workflow</td>
<td></td>
<td>√</td>
<td>~</td>
</tr>
</tbody>
</table>
take away
big data $\gg$ little data

transformational factors

situation factors

result

little brain

problem

www.cs.ubc.ca/~murphy
www.tasktop.com