If Software Could Talk...

... why changes are made.

Audris Mockus
audris@research.bell-labs.com
http://www.bell-labs.com/~audris
Motivation

- Software databases
  - always there
    - large
    - uniform over time
- Empirically based understanding - lacking
  - GQM experiments - difficult
  - not usable as project tracking tools
  - cannot be widely deployed
Approach

- Any project has VCS data
  - Derive/augment data to:
    - explain variability in quality, effort, interval
  - Validate
    - survey, other products
  - Apply the results for
    - process understanding (org., decay, infrast.)
    - problem localization and tracking
Outline

- Switching software change data
  - Classification of changes
    - automatic algorithm
      - experimental verification
  - Size and difficulty of changes
- Other applications
  - org. theory, code decay, expertise localization, effort estimation
Software projects

- two decades of development
  - distributed/real-time software
    - 8x more complex than application software (SEI)
  - scale:
    - 100 million lines of code
    - 100 thousand pages documentation
    - 20 supported versions
- sophisticated development process
- thousands of software engineers
How Code Evolves

- By adding and deleting line blocks

before:  after:

```c
// initialize
int i=0;    int i=0;
while (i++)  while (i++ < N)
    read (x); read(x);
```

- one line deleted
- two lines added
- two lines unchanged
Why study changes

- Reflect relationships between
  - requirements and design
    - technology and implementation
    - personnel (organization)
    - time (evolution of the system)
  - Practical
    - always documented by VCS
    - results have wide applicability
Any VCS Records:

- **Change** - added and deleted lines
  - **Who** - login, organization
  - **When** - date and time
  - **Description** - line of text
- **Available data:**
  - ~100M lines, ~3M changes, ~5K logins, 20Gb
  - ~30 products (select one)
No VCS Records for:

- Why
  - fix, new, improve, ...

- How difficult
  - effort, interval, complexity

- Will it cause fault in the future
  - estimate fault potential
Why Change?

- Primary reasons for maintenance activities
  - corrective: fix faults
  - adaptive: add features

- How those reasons relate to:
  - interval, effort, quality
  - developer, size
  - location, time
How to extract

- Use change description line
  - extract frequent keywords
  - classify keywords (fix, new, add, etc.)
    - discover new types
      - perfective - code cleanup
      - inspection - code inspection suggestions
  - verify on sample abstracts
- keyword -> purpose of the change
- iterate
<table>
<thead>
<tr>
<th>Keywords</th>
<th>Adaptive</th>
<th>Corrective</th>
<th>Perfective</th>
<th>Inspection</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>add, new, create,</td>
<td>fix, bug, error,</td>
<td>cleanup, remove, clear, unneeded, flex name</td>
<td>code review, inspection, rework, walkthrough</td>
</tr>
<tr>
<td></td>
<td>initial coding, modify, update</td>
<td>problem, incorrect,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>must, needs</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Adaptive:**
- add
- new
- create
- initial coding
- modify
- update

**Corrective:**
- fix
- bug
- error
- problem
- incorrect
- must
- needs

**Perfective:**
- cleanup
- remove
- clear
- unneeded
- flex name

**Inspection:**
- code review
- inspection
- rework
- walkthrough
Proportions

- Why:
  - add new functionality - 45%
  - fix faults (bug) - 34%
  - cleanup/restructure - 4%
  - code inspection - 5%
  - unclassified - 12%
Is it right?

Survey:
- 2+5 developers (>9 years experience)
  - 20+150 changes (< 2 years old)
    - ~ equal numbers for different types
      - small percent of all changes developers did

Questions
- Type: bug, new, cleanup, inspection
- Difficulty: Easy, Medium, Hard
Results

- Unclassified changes are mostly bug fixes
  - Almost perfect match
- Inspection changes are easiest to detect

<table>
<thead>
<tr>
<th>Dev./Auto</th>
<th>Corrective</th>
<th>Adaptive</th>
<th>Perfective</th>
<th>Inspection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corrective</td>
<td>35</td>
<td>10</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>Adaptive</td>
<td>11</td>
<td>23</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Perfective</td>
<td>10</td>
<td>8</td>
<td>27</td>
<td>9</td>
</tr>
<tr>
<td>Inspection</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>21</td>
</tr>
</tbody>
</table>
Will it work?

- Other Product
  - 2 X size and five years older
    - different functionality
    - different organization

- Tool
  - the same classification (no manual input)

- Results
  - very similar purpose profiles
Why changes are difficult?

$$\text{Difficulty} \sim \text{Fix} + \text{Developer} + \text{Size} + \log(\text{Interval})$$

- Fixes are hard - no matter size
  - Interval barely important adjusted for size
  - Type and size are more important than developer

<table>
<thead>
<tr>
<th>Factor</th>
<th>DF</th>
<th>SS</th>
<th>p-val</th>
<th>effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fix</td>
<td>1</td>
<td>12</td>
<td>0</td>
<td>+</td>
</tr>
<tr>
<td>Size</td>
<td>1</td>
<td>7</td>
<td>0</td>
<td>+</td>
</tr>
<tr>
<td>Developer</td>
<td>6</td>
<td>10</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>log(interval)</td>
<td>1</td>
<td>1.4</td>
<td>.027</td>
<td>+</td>
</tr>
<tr>
<td>Cleanup</td>
<td>1</td>
<td>.6</td>
<td>.12</td>
<td>+</td>
</tr>
<tr>
<td>Residuals</td>
<td>159</td>
<td>45</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Is change difficult?

- Difficult
  - more than 2 files touched, many delta, fault fix
    - Frequently repeated, predominant
      - more 100 times, at least 30% of the time
    - Are different parts equally difficult?
    - Are changes becoming harder over time?
    - Where to reengineer the code?
Summary

- Algorithm to extract purpose
  - automatic
    - validated by survey, on other product
- 4 types of changes discovered
  - different size, interval profiles
- Relationships
  - difficult - type
  - size, interval type
Summary

- Can VCS be used to find out:
  - why change is made
    - why change is difficult

- Obtain essential properties of changes
  - Data source available for all SW projects
  - Non-intrusive data collection
  - Methodology to describe software projects

- Potential to predict the impact of:
  - organizational (team size)
  - process (code inspections)
  - technology (compilers, computer languages)
Future & Current Work

- Refine classification
  - detail - type of fix - overflow, deadlock, ...
    - domain - HW/SW, phase
- Utilize other databases
  - financial support system - effort
  - customer tracking - serious faults
- VCS enhancement tools
  - problem localization, project status
Fault Potential

- Do past changes predict future faults
  - predict proportion of faults
    - in a two year period
      - for 88 modules
    - numbers, sizes, age of changes
  - Best predictor:
    - past number of faults
    - but NOT: complexity, connectivity, #authors
Can developers know:

- Which subsystems/modules are hard?
  - What types of changes are frequent?
  - Who writes the most code?

- Access platform goals:
  - standard Netscape interface
    - no software/data to install
    - point and click
Link: Developer activity

All icons (developers) are ordered according to login and then laid out in an upside-down text flow order (from left to right and then one row up, etc.). Some of the icons are much smaller than others. Those icons represent logins that did not make many changes. For each icon to take full space use "Local Scaling".
The numbers of changes are square rooted before display. To choose a different transformation use:

For closer inspection the view can be panned (drag left mouse button) and zoomed (shift-drag left mouse button). Icon sizes can be adjusted using scrollbar on the left. The style of iconic representation can be one of the following styles: Rays. Choice "Stars" is similar to the current choice "Rays". Choice "Parallel" would show the changes as a time series.
Table view
<table>
<thead>
<tr>
<th>Cases</th>
<th>0</th>
<th>b</th>
<th>c</th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td>tom</td>
<td>577</td>
<td>628</td>
<td>63</td>
<td>1169</td>
</tr>
<tr>
<td>ccc</td>
<td>102</td>
<td>127</td>
<td>90</td>
<td>1040</td>
</tr>
<tr>
<td>emilyr</td>
<td>427</td>
<td>194</td>
<td>15</td>
<td>968</td>
</tr>
<tr>
<td>cuervo</td>
<td>1602</td>
<td>1090</td>
<td>452</td>
<td>772</td>
</tr>
<tr>
<td>dipietro</td>
<td>817</td>
<td>1583</td>
<td>218</td>
<td>701</td>
</tr>
<tr>
<td>atang</td>
<td>153</td>
<td>1</td>
<td>0</td>
<td>580</td>
</tr>
<tr>
<td>liuzzo</td>
<td>376</td>
<td>252</td>
<td>111</td>
<td>559</td>
</tr>
<tr>
<td>bhp</td>
<td>1101</td>
<td>469</td>
<td>137</td>
<td>554</td>
</tr>
<tr>
<td>jedavis</td>
<td>97</td>
<td>117</td>
<td>5</td>
<td>499</td>
</tr>
<tr>
<td>stanza</td>
<td>273</td>
<td>116</td>
<td>361</td>
<td>475</td>
</tr>
<tr>
<td>sae</td>
<td>462</td>
<td>1651</td>
<td>85</td>
<td>454</td>
</tr>
<tr>
<td>sunu</td>
<td>1618</td>
<td>150</td>
<td>94</td>
<td>449</td>
</tr>
<tr>
<td>cvbl</td>
<td>333</td>
<td>403</td>
<td>89</td>
<td>448</td>
</tr>
<tr>
<td>keml</td>
<td>393</td>
<td>130</td>
<td>47</td>
<td>437</td>
</tr>
<tr>
<td>tich</td>
<td>541</td>
<td>159</td>
<td>298</td>
<td>432</td>
</tr>
<tr>
<td></td>
<td>686</td>
<td>285</td>
<td>95</td>
<td>400</td>
</tr>
<tr>
<td>Cases</td>
<td>0</td>
<td>b</td>
<td>c</td>
<td>n</td>
</tr>
<tr>
<td>---------</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>dlo</td>
<td>373</td>
<td>281</td>
<td>625</td>
<td>236</td>
</tr>
<tr>
<td>doris</td>
<td>1375</td>
<td>447</td>
<td>579</td>
<td>388</td>
</tr>
<tr>
<td>kjs</td>
<td>635</td>
<td>1225</td>
<td>521</td>
<td>184</td>
</tr>
<tr>
<td>cuervo</td>
<td>1602</td>
<td>1090</td>
<td>452</td>
<td>772</td>
</tr>
<tr>
<td>bmary</td>
<td>865</td>
<td>376</td>
<td>445</td>
<td>30</td>
</tr>
<tr>
<td>senior</td>
<td>723</td>
<td>149</td>
<td>386</td>
<td>243</td>
</tr>
<tr>
<td>stanza</td>
<td>273</td>
<td>116</td>
<td>361</td>
<td>475</td>
</tr>
<tr>
<td>rosson</td>
<td>383</td>
<td>140</td>
<td>317</td>
<td>19</td>
</tr>
<tr>
<td>mts</td>
<td>54</td>
<td>84</td>
<td>312</td>
<td>320</td>
</tr>
<tr>
<td>tuch</td>
<td>541</td>
<td>159</td>
<td>298</td>
<td>432</td>
</tr>
<tr>
<td>kjs28</td>
<td>573</td>
<td>1016</td>
<td>295</td>
<td>376</td>
</tr>
<tr>
<td>scooby</td>
<td>433</td>
<td>1144</td>
<td>275</td>
<td>151</td>
</tr>
<tr>
<td>fenn</td>
<td>301</td>
<td>207</td>
<td>269</td>
<td>194</td>
</tr>
<tr>
<td>shp</td>
<td>165</td>
<td>406</td>
<td>228</td>
<td>180</td>
</tr>
<tr>
<td>dipietro</td>
<td>817</td>
<td>1583</td>
<td>218</td>
<td>701</td>
</tr>
</tbody>
</table>