We must turn our attention towards the people that build software!
Which should you test most?

Past Failures
Complexity Metrics

- Measure “complexity” of the source code:
  - #Lines
  - #Classes
  - #Parameters
- Higher metric = greater complexity

McCabe Metrics

- Measure complexity of control flow
- \( V(G) = e - n + 2p \)
  - \( e \): #edges
  - \( n \): #statements
  - \( p \): #entries

Maintainability Index

\[
\text{Maintainability} = 171 - 5.2 \ln(V) - 0.23V(G) - 16.2 \ln(L) \\
+ 50 \sin \left( \sqrt{2.4C} \right)
\]

- Size of vocabulary
- McCabe complexity
- Percentage of comment lines
- code lines
Complexity Metrics

What can we use to predict failures?

A Combined Approach

- Collected failures occurring in the field within 6 months after release
- Mapped failures back to fixes and thus defects in modules (binaries)
- We can tell how failure-prone a module is
A Combined Approach

Past failures

Metrics

<table>
<thead>
<tr>
<th>#Lines</th>
<th>#Vars</th>
<th>#Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Params</td>
<td>#Reads</td>
<td>#Writes</td>
</tr>
<tr>
<td>#Arcs</td>
<td>#Blocks</td>
<td>McCabe</td>
</tr>
</tbody>
</table>
| Fan In | Fan Out | ...

A combined approach

Past failures

Metrics

<table>
<thead>
<tr>
<th>#Lines</th>
<th>#Vars</th>
<th>#Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Params</td>
<td>#Reads</td>
<td>#Writes</td>
</tr>
<tr>
<td>#Arcs</td>
<td>#Blocks</td>
<td>McCabe</td>
</tr>
</tbody>
</table>
| Fan In | Fan Out | ...

Correlation

McCabe ~ Failures
Predicting failure

Correlation \[ \text{McCabe} \sim \text{Failures} \]

Projects Researched

- Internet Explorer 6
- IIS Server
- Windows Process Messaging
- DirectX
- NetMeeting

>1,000,000 Lines of Code
Research Questions

- Do metrics correlate with failures?
- Is there a set of metrics that fits all projects?
- Can we predict failure-prone modules?

Do metrics correlate with failures?

<table>
<thead>
<tr>
<th>Project</th>
<th>Metrics correlated w/ failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>#Classes and 5 derived</td>
</tr>
<tr>
<td>B</td>
<td>almost all</td>
</tr>
<tr>
<td>C</td>
<td>all except \textit{MaxInheritanceDepth}</td>
</tr>
<tr>
<td>D</td>
<td>only #Lines</td>
</tr>
<tr>
<td>E</td>
<td>#Functions, #Arcs, Complexity</td>
</tr>
</tbody>
</table>

YES
Is there a set of metrics that fits all projects?

<table>
<thead>
<tr>
<th>Project</th>
<th>Metrics correlated w/ failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>#Classes and 5 derived</td>
</tr>
<tr>
<td>B</td>
<td>almost all</td>
</tr>
<tr>
<td>C</td>
<td>all except MaxInheritanceDepth</td>
</tr>
<tr>
<td>D</td>
<td>only #Lines</td>
</tr>
<tr>
<td>E</td>
<td>#Functions, #Arcs, Complexity</td>
</tr>
</tbody>
</table>

Is there a set of metrics that fits all projects?

Can we predict failure-prone modules?

- Basic idea: Combine metrics
- Give most weight to most predictive metrics
- Problem: Metrics are intercorrelated
- Solution: Principal Component Analysis (PCA)
Predictive Power

- From the principal components, we can build regression models
- These can be fed with metrics to predict the defect likelihood
- Modules can be ranked according to likelihood

A Ranking

- 1/3 of the modules
- ranked according to predictor built from 2/3 of the modules
- can be evaluated against actual ranking
Can we predict failure-prone modules?

<table>
<thead>
<tr>
<th>Project</th>
<th>#Components</th>
<th>$R^2$ value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>9</td>
<td>0.741</td>
</tr>
<tr>
<td>B</td>
<td>6</td>
<td>0.779</td>
</tr>
<tr>
<td>C</td>
<td>7</td>
<td>0.579</td>
</tr>
<tr>
<td>D</td>
<td>7</td>
<td>0.684</td>
</tr>
<tr>
<td>E</td>
<td>5</td>
<td>0.919</td>
</tr>
</tbody>
</table>
Can we predict failure-prone modules?

YES

<table>
<thead>
<tr>
<th>Project</th>
<th>#Components</th>
<th>$R^2$ value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>9</td>
<td>0.741</td>
</tr>
<tr>
<td>B</td>
<td>6</td>
<td>0.779</td>
</tr>
<tr>
<td>C</td>
<td>7</td>
<td>0.579</td>
</tr>
<tr>
<td>D</td>
<td>7</td>
<td>0.684</td>
</tr>
<tr>
<td>E</td>
<td>5</td>
<td>0.919</td>
</tr>
</tbody>
</table>

Are predictors applicable across projects?

<table>
<thead>
<tr>
<th>Project</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>•</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>B</td>
<td>no</td>
<td>•</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>C</td>
<td>no</td>
<td>yes</td>
<td>•</td>
<td>no</td>
<td>(yes)</td>
</tr>
<tr>
<td>D</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>•</td>
<td>no</td>
</tr>
<tr>
<td>E</td>
<td>no</td>
<td>no</td>
<td>(yes)</td>
<td>no</td>
<td>•</td>
</tr>
</tbody>
</table>

Are predictors applicable across projects?

DEPENDS
Consequences

• DO NOT use complexity metrics without validating them for your project
• DO consider using validated metrics to identify failure-prone components

Future Work

<table>
<thead>
<tr>
<th>More metrics</th>
<th>More projects</th>
</tr>
</thead>
<tbody>
<tr>
<td>More automation</td>
<td>More data</td>
</tr>
</tbody>
</table>

Hatari
What is it that makes some components buggier than others?

Is it the Developers?

Does experience matter? Bug density correlates with experience!

Is it History?

I found lots of bugs here. Will there be more? Yes! (But where did these come from?)
How about metrics?

Do code metrics correlate with bug density?
Sometimes!

Ah! Language features?
Are gotos harmful?
No correlation!

Ok. Problem Domain?
Which tokens do matter?
import • extends • implements
Eclipse Imports

import org.eclipse.jdt.internal.compiler.lookup.*;
import org.eclipse.jdt.internal.compiler.*;
import org.eclipse.jdt.internal.compiler.ast.*;
import org.eclipse.jdt.internal.compiler.util.*;
... import org.eclipse.pde.core.*;
import org.eclipse.jdt.jface.wizard.*;
import org.eclipse.ui.*;

14% of all components importing \texttt{ui} show a post-release defect

14% of all components importing \texttt{pde.core} show a post-release defect

Joint work with Adrian Schröter • Tom Zimmermann

Eclipse Imports

Correlation with failure

Correlation with success

Which parts should be tested most?

Correlation with success

Which parts should be tested most?
Predicting failure-prone packages

- Similar to Microsoft study, but rely on imports rather than metrics
- Base: Eclipse with bug and version database (Bugzilla, CVS)
- 36% of all packages had post-release defects
- Prediction using support vector machine

Results

~300 Packages

- Top 5%
- 90% defect
- 90% no defect

Prediction at design time!

And what else?

But are there domain-independent properties?
Summary

⭐ Metrics correlate with failures
⭐ There are no universal metrics that would be applicable to all projects
⭐ We can predict failure-prone modules
⭐ Software archives tell a lot about real errors
⭐ Abundance of data for mining and evaluation

Download at http://www.st.cs.uni-sb.de/softevo/