Bug Prediction

SW Wartung und Evolution

Emanuel Giger
Software has Bugs!
Software has Bugs!
Software has Bugs!
Software has Bugs!
Software has Bugs!
Software has Bugs!

Bugs! Bugs! Bugs! Bugs! Bugs!

Bugs! Bugs! Bugs! Bugs! Bugs!

Friday, May 25, 2012
Quality Assurance (QA)...

...is limited by time and money
Quality Assurance (QA)...

...is limited by time and money

Spend resources with maximum efficiency!
Focus on the components that fail the most!
Bug Prediction Models

Bug Prediction

Change Metrics
- Previous Bugs
- Code Churn
- Fine-Grained Source Changes

Code Metrics
- Function Level Metrics
- OO-Metrics

Organizational Metrics
- Team Structure
- Contribution Structure

Method-Level Bug Prediction
Bug Prediction Models

- Change Metrics
  - Previous Bugs
  - Code Churn
  - Fine-Grained Source Changes
  - Method-Level Bug Prediction

- Code Metrics
  - Function Level Metrics
  - OO-Metrics

- Organizational Metrics
  - Team Structure
  - Contribution Structure
Code Metrics

- Directly calculated on the code itself
- Measure size and complexity of the code
Code Metrics

• Directly calculated on the code itself
• Measure size and complexity of the code
Code Metrics

- Directly calculated on the code itself
- Measure size and complexity of the code
Code Metrics

- Directly calculated on the code itself
- Measure size and complexity of the code
Code Metrics

- Directly calculated on the code itself
- Measure size and complexity of the code

Lines of Code

Dependency

McCabe

Inheritance
Bug Prediction Setup

Eclipse
Bug Prediction Setup

Eclipse → Code Metrics & Bug Data
Bug Prediction Setup

Eclipse → Code Metrics & Bug Data → Random Forest
Bug Prediction Setup

Eclipse → Code Metrics & Bug Data

Code Metrics & Bug Data → Random Forest

Random Forest → X-Validation
Bug Prediction Setup

- Eclipse
- Code Metrics & Bug Data
- Bug-Prone
- Not Bug-Prone
- Random Forest
- X-Validation

Friday, May 25, 2012
Bug Prediction Models

Change Metrics
- Previous Bugs
- Code Churn
- Fine-Grained Source Changes

Bug Prediction
- Code Metrics
  - Function Level Metrics
  - OO-Metrics
- Organizational Metrics
  - Team Structure
  - Contribution Structure

Method-Level Bug Prediction
Revisions

Commits to version control systems

Coarse-grained

Files are the units of change
There is more than just a file revision.
Revisions

There is more than just a file revision
Revisions

There is more than just a file revision
Revisions

There is more than just a file revision
Revisions

There is more than just a file revision

```java
private IStructureComparator fStructureComparator;

public boolean setInput(ITypedElement newInput, boolean force) {
    boolean changed = false;
    if (force || newInput != fInput) {
        removeDocumentRangeUpdaters();
        if (fInput instanceof IContentChangeNotifier)
            ((IContentChangeNotifier)fInput).removeContentChangeListener(fContentChangeListener);
        fInput= newInput;
        if (fInput == null) {
            if (fStructureComparator instanceof IDisposable) {
                IDisposable disposable = (IDisposable)fStructureComparator;
                disposable.dispose();
            }
            fStructureComparator= null;
        } else {  // else { fInput instanceof IContentChangeNotifier }  
            fStructureComparator= (IStructureComparator)fInput;
            addedContentChangeListener(fContentChangeListener);
        }
    } return changed;
}

/**
 * Remove any document range updaters that were registered against the document.
 */
private void removeDocumentRangeUpdaters() {
    if (fStructureComparator instanceof IDocumentRange) {
        IDocument doc = ((IDocumentRange)fStructureComparator).getDocument();
        try {
            // do the unregister
        } finally {
            // unregister
        }
    }
}

public IStructureComparator getStructureComparator() {
    return fStructureComparator;
}

public void refresh() {
    IStructureComparator oldComparator = fStructureComparator;
    fStructureComparator= createStructure();
    if (fStructureComparator != null)
        addedContentChangeListener(fContentChangeListener);
}
```
Revisions

There is more than just a file revision

```java
private IStructureComparator fStructureComparator;
public boolean setInput(ITypedElement newInput, boolean force) {
    boolean changed = false;
    if (force || newInput != fInput) {
        removeDocumentRangeUpdaters();
        if (fInput instanceof IContentChangeNotifier)
            ((IContentChangeNotifier)fInput).removeContentChangeEventListener(fContentChangeEventListener);
        fInput = newInput;
    }
    // More code...
```

```java
private IStructureComparator fStructureComparator;
public boolean setInput(ITypedElement newInput, boolean force) {
    boolean changed = false;
    if (force || newInput != fInput) {
        removeDocumentRangeUpdaters();
        if (fInput instanceof IContentChangeNotifier)
            ((IContentChangeNotifier)fInput).removeContentChangeEventListener(fContentChangeEventListener);
        fInput = newInput;
    }
    // More code...
```
Revisions

There is more than just a file revision

```csharp
private IStructureComparator fStructureComparator;

public boolean setInput(ITypedElement newInput, boolean force) {
    boolean changed = false;
    if (force || newInput != fInput) {
        removeDocumentRangeUpdaters();
        if (fInput instanceof IContentChangeNotifier) {
            ((IContentChangeNotifier)fInput).removeContentChangeListener(fContentChangeListener);
        }
        fInput = newInput;
        if (fInput == null) {
            fStructureComparator = null;
            return false;
        }
        fStructureComparator = null;
        if (fInput instanceof IContentChangeNotifier) {
            ((IContentChangeNotifier)fInput).addContentChangeListener(fContentChangeListener);
        }
        return changed;
    }
}
```

```csharp
private ITypedElement fInput;
private IStructureComparator fStructureComparator;

public boolean setInput(ITypedElement newInput, boolean force) {
    boolean changed = false;
    if (force || newInput != fInput) {
        removeDocumentRangeUpdaters();
        if (fInput instanceof IContentChangeNotifier) {
            ((IContentChangeNotifier)fInput).removeContentChangeListener(fContentChangeListener);
        }
        fInput = newInput;
        if (fInput == null) {
            refresh();
            changed = true;
        } else {
            fStructureComparator = null;
            if (fInput instanceof IContentChangeNotifier) {
                ((IContentChangeNotifier)fInput).addContentChangeListener(fContentChangeListener);
            }
            return changed;
        }
    }
}
```

```csharp
/**
 * Remove any document range updaters that were registered against the document.
 */
private void removeDocumentRangeUpdaters() {
    if (fStructureComparator instanceof IDocumentRange) {
        IDocument doc = ((IDocumentRange)fStructureComparator).getDocument();
        try {
            doc.removeDocumentRangeUpdaters(fStructureComparator);
            fStructureComparator = null;
        } finally {
            fStructureComparator = null;
        }
    }
}
```

```csharp
public void refresh() {
    IStructureComparator oldComparator = fStructureComparator;
    fStructureComparator = createStructure();
}
```
Code Changes

Revisions
- Commits to version control systems
- Coarse-grained
- Files are the units of change

Code Churn
- Textual UnixDiff between 2 File Versions
- Ignores the structure of code
- No change type information
- Includes textual changes
Code Churn

Does not reflect the type and the semantics of source code changes
Code Changes

Revisions
- Commits to version control systems
- Coarse-grained
  - Files are the units of change

Code Churn
- Textual UnixDiff between 2 File Versions
  - Ignores the structure of code
  - No change type information
  - Includes textual changes

Fine-Grained Changes
- Compares 2 versions of the AST of source code
  - Very fine-grained
  - Change type information
  - Captures all changes
Code Changes

Revisions

Commits to version control systems
Coarse-grained
Files are the units of change

Code Churn

Textual UnixDiff between 2 File Versions
Ignores the structure of code
No change type information
Includes textual changes

Fine-Grained Changes

Compares 2 versions of the AST of source code
Very fine-grained
Change type information
Captures all changes

¹[Fluri et al. 2007, TSE]
Account.java 1.5

IF "balance > 0"

THEN

MI

"withDraw(amount);"
Fine-grained Changes

Account.java 1.5

IF "balance > 0"

THEN

MI "withDraw(amount);"

Account.java 1.6

IF "balance > 0 && amount <= balance"

THEN

MI "withDraw(amount);"

ELSE

MI notify();
Fine-grained Changes

Account.java 1.5

IF "balance > 0"

THEN

MI

"withDraw(amount);"

Account.java 1.6

IF "balance > 0 && amount <= balance"

THEN

MI

"withDraw(amount);"

ELSE

MI

MI

notify();

1x condition change, 1x else-part insert, 1x invocation statement insert
Fine-grained Changes

Account.java 1.5

IF "balance > 0"
THEN
MI
"withDraw(amount);"

Account.java 1.6

IF "balance > 0 && amount <= balance"
THEN
MI
"withDraw(amount);"
ELSE
MI
notify();

1x condition change, 1x else-part insert, 1x invocation statement insert
Fine-grained Changes

Account.java 1.5

IF "balance > 0"
THEN
MI
"withDraw(amount);"

Account.java 1.6

IF "balance > 0 && amount <= balance"
THEN
MI
"withDraw(amount);"
ELSE
MI
notify();

More accurate representation of the change history

1x condition change, 1x else-part insert, 1x invocation statement insert
Method-Level Bug Prediction

class 1  class 2  class 3  ...  class n
Method-Level Bug Prediction

11 methods on average
Method-Level Bug Prediction

11 methods on average
4 are bug prone
Method-Level Bug Prediction

11 methods on average
4 are bug prone

Retrieving bug-prone methods saves manual inspection steps and improves testing effort allocation
Method-Level Bug Prediction

Retrieving bug-prone methods saves manual inspection steps and improves testing effort allocation.
Bug Prediction Models

- Bug Prediction
  - Change Metrics
    - Previous Bugs
    - Code Churn
    - Fine-Grained Source Changes
  - Code Metrics
    - Function Level Metrics
    - OO-Metrics
  - Organizational Metrics
    - Team Structure
    - Contribution Structure

Method-Level Bug Prediction
Bug Prediction Models

Using the Gini Coefficient for Bug Prediction
Gini Coefficient

- The Lorenz curve plots the cumulative % of the total participation against the cumulative % of the population.
- Gini Coefficient summarizes the curve in a number.
Gini Coefficients are reported in %


Friday, May 25, 2012
Income Distribution

Gini Coefficients are reported in %


Friday, May 25, 2012
What about Software?
What about Software?

Developers = Population
What about Software?

Files = Assets

Developers = Population
What about Software?

Developers = Population

Changing a file = “being owner”

Files = Assets
What about Software?

How are changes of a file distributed among the developers and how does this relate to bugs?

Developers = Population

Changing a file = “being owner”

Files = Assets

How are changes of a file distributed among the developers and how does this relate to bugs?
Lorenz Curve of Eclipse Resource

Cumulative % of Developer Population

Cumulative % of Revisions

A

B

Friday, May 25, 2012
Gini Coefficient

\[ G = \frac{A}{A + B} \]
Study

- Eclipse Dataset
- Avg. Gini coefficient is 0.9
- Namibia has a coefficient of 0.7
- Negative Correlation of ~-0.55
- Can be used to identify bug-prone files
• Eclipse Dataset
• Avg. Gini coefficient is 0.9
• Namibia has a coefficient of 0.7
• Negative Correlation of ~-0.55
• Can be used to identify bug-prone files

The more changes of a file are done by a few dedicated developers the less likely it will be bug-prone!
Economic Phenomena

- Economic phenomena of code ownership
- Economies of Scale (Skaleneffekte)
- I’m an expert (in-depth knowledge)
- Profit from knowledge
Economic Phenomena

- Economic phenomena of code ownership
- Economies of Scale (Skaleneffekte)
- I’m an expert (in-depth knowledge)
- Profit from knowledge

Costs to acquire knowledge can be split, e.g., among several releases if you stay with a certain component
Diseconomies of Scale

• Negative of effect of code ownership?
• Loss of direction and co-ordination
• Are we working for the same product?
Another Phenomena

- Economies of Scope (Verbundseffekte)
- Profiting from breadth-knowledge
- Knowledge of different components helps in co-ordinating
- Danger of bottlenecks!
Implications & Conclusions

- How much code ownership & expertise?
- What is your bus number?
- What is better? In-depth- or breadth-knowledge?
- What’ is the optimal team size?