Wei Yang
A Grey-Box Approach for Automated GUI-Model Generation of Mobile Applications

**a1**: Toggle exclude tax rate option.
**a2**: Toggle round up option.
AT-EASE: A Tool for Early and Quick Usability Evaluation of Smartphone Application
WHYPER: Towards Automating Risk Assessment of Mobile Applications
UIUC SE Seminar

Xusheng Xiao

xxiao2@ncsu.edu / xxiao9@illinois.edu
Cooperative Testing and Analysis via Informed Decision Making

• Structural Test Generation
  – Not What coverage
  – Why and How such coverage
    [ICSE 2011, ICSE Demo 2011, ASE 2013]

• Security analysis on mobile platforms
  – Not What permission
  – Why and How to use permission
    [ASE 2012]

• Performance analysis on GUI applications
  – Not What expensive
  – Why and How expensive
    [ISSTA 2013]
Text Analytics

• Model Extraction to Assist Verification
  – Access Control Policy  [FSE 2012]
  – Pre/Post Condition  [ICSE 2012]

• Sentence Classification to Assist Security Analysis
  – Android Descriptions for Permissions  [USENIX Security 2013]
Research Interests

Alex Gyori
1st Year PhD student

Co-advised by: Darko Marinov and Danny Dig
Past: Refactoring Imperative to Functional

```java
for ( Block b : blocks) {
    if (b.isVisible()) {
        int area = b.getArea();
        accumulator += area;
    }
}

accumulator += blocks.stream().parallel().
    filter(b -> b.isVisible())
    .map(b -> b.getArea())
    .reduce(0, Integer::sum);
```

Want to know more? Read our FSE'13 or ICSE'13 papers: mir.cs.Illinois.edu/~gyori

OR ASK!
Present: Bugs

• Testing in the context of Software Evolution:
  • Regression Testing
  • Test Selection
  • Test Prioritization
Future: Have fun!?! Find Thesis topic!

- Anything to do with bugs - testing (functional, security, concurrency)
- Program transformations/Refactorings (parallelism)
- Using static or dynamic analysis, model checking, SMT to solve SE problems
- Just Hacking random code!

If you find anything on the slides that sounds related to your interests we should talk.
Collecting Java Concurrency Bug Benchmark
Ziyi Lin
Research goal

- Collecting a set of Java programs
- With known concurrency bugs
- To evaluate concurrency bug detection tools
Research stages

- Choose proper source code repository.
- Find bugs
- Set up profile for each bug
- Verify benchmark by existing concurrency bug detection tools
- Maintain and update benchmarks
Choose source code

- Representative: real
- Diverse: bug patterns, code sizes,…
- Accessible: open source
- Fair: not bias
Locating bugs

- Most important part.
- Find as many concurrency bugs as possible.
  - From bug management systems
  - Run on existing detection tools
  - Generate new test cases to find more bugs
Setting up profiles

- Description of the bug
- How to trigger the bug
- How much time to discover the bug
- ...
Maintenance

- Add new into benchmark set
- Update bug profiles
Start from existing programs

- S.I.R.
- Yaniv Eytani [ICST08]
- ReEx, IMunit

<table>
<thead>
<tr>
<th>Program</th>
<th>System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collections</td>
<td>IMunit</td>
</tr>
<tr>
<td>Jboss-Cache</td>
<td>IMunit</td>
</tr>
<tr>
<td>Lucene</td>
<td>IMunit</td>
</tr>
<tr>
<td>Mina</td>
<td>IMunit, ReEx</td>
</tr>
<tr>
<td>Pool</td>
<td>IMunit, ReEx</td>
</tr>
<tr>
<td>Sysunit</td>
<td>ReEx</td>
</tr>
<tr>
<td>JSR-166 TCK</td>
<td>ReEx</td>
</tr>
<tr>
<td>Airline</td>
<td>ReEx</td>
</tr>
<tr>
<td>Allocation</td>
<td>ReEx</td>
</tr>
<tr>
<td>BoundedBuffer</td>
<td>ReEx</td>
</tr>
<tr>
<td>BubbleSort</td>
<td>ReEx</td>
</tr>
<tr>
<td>Deadlock</td>
<td>ReEx</td>
</tr>
<tr>
<td>ReadersWriters</td>
<td>ReEx</td>
</tr>
<tr>
<td>ReplWorkers</td>
<td>ReEx</td>
</tr>
</tbody>
</table>
About me

Cosmin Rădoi
Last year

Experience

concurrent

program

practical

race
detection

paralel

Java

WALA

Scala

C#

c/c++

Ruby

web

Java

C#

Ruby

web

Experience

Interest

concurrent

cLOUD

transformation

scalable

race
detection

parallel

Java

C#

Ruby

web

Java

C#

Ruby

web

Experience

Interest

concurrent

cLOUD

transformation

scalable

race
detection

parallel

Java

C#

Ruby

web

Java

C#
Projects

- imperative sequential $\rightarrow$ map-reduce
  with Manu Sridharan, Stephen Fink, Rodric Rabbah

- is there latent parallelism in emerging JavaScript applications?
  with Stephan Herhut, Jaswanth Sreeram, Danny Dig

- static race detection for Java parallel loops
  ISSTA ’13 (with Danny Dig)

- parallel inclusion-based pointer analysis using actors
  with Semih Okur, Gul Agha
map-reduce transformation example

```java
for (int i=0; i<n; i++) {
    b[i] = a[i] + i;
}

b = a . map { (i, v) => (i, v + i) }
```
RESEARCH INTERESTS

Semih Okur, 3rd year PhD student

Parallel & Asynchronous programming,
Mobile software,
Source-code transformations and static analysis,
Performance profilers
Source-code mining and empirical studies
HOW DO DEVELOPERS USE PARALLEL LIBRARIES?

Semih Okur & Danny Dig

First large-scale empirical study for parallel libraries. Analyzed 655 C# applications 17.6M SLOC by 1609 programmers.

We answered 8 research questions related to adoption, frequently (mis)used constructs, and patterns.

www.LearnParallelism.NET
A STUDY AND TOOLKIT FOR ASYNCHRONOUS PROGRAMMING IN C#

Semih Okur & David Hartveld & Danny Dig & Arie van Deursen

Analyzed 1378 Windows Phone Apps to answer:
Q1) How do developers use asynchronous programming?
Q2) To what extent do developers misuse async/await?

www.LearnAsync.NET
ASYNCIFIER AND CORRECTOR

Developers still use callback-based asynchronous idioms. Developers have to refactor!

1) Asyncifier: We implemented a refactoring tool for 4 common misuses. For instance, 14% of methods do not need to use async/await keywords. 20% of methods do not follow an important practice.

2) Corrector: We implemented a tool to fix these misuses.
MORE RESPONSIVE MOBILE APPS

Developers misuse async/await, causing many performance bugs.

Goals:

1) Fix the specific misuses
2) Implement a profiler to dynamically analyze UI thread.
REFACTORING FOR SCALABILITY

Developers still use old-style multithreading constructs which hurt the performance, scalability, and readability

Goals:
1) Transform Thread and blocking synchronization constructs to Task and non-blocking ones.
Yu Lin
Co-advised by Darko and Danny
Past Research

• Coverage Driven Testing of Akka Actor programs
  – Generating and exploring representative interleavings based on coverage criteria inherited from bug patterns

• Check-then-Act Misuse of Java Concurrent Collections
  – Detecting atomicity violations bugs when using concurrent collections

• Evaluating Machine-Independent Metrics for State-Space Exploration
  – Evaluate the correlation of machine-independent metrics with real time
Ongoing Research

• Automatic Distributed Service Orchestration Testing and Tuning

• Service-Level Testing for Distributed Services
  – Testing the correctness/performance in the presence of environmental disturbances
  – E.g., network failures/delays, VM crashes, VM resource contentions
final class InputStreamImageData extends ZLAndroidImageData {
    private final ZLSingleImage myImage;

    InputStreamImageData(ZLSingleImage image) {
        myImage = image;
    }

    protected Bitmap decodeWithOptions(BitmapFactory.Options options) {
        final InputStream stream = myImage.inputStream();
        if (stream == null) {
            return null;
        }

        final Bitmap bmp = BitmapFactory.decodeStream(stream, new Rect(), options);
        try {
            stream.close();
        } catch (IOException e) {
        }
        return bmp;
    }
}
public class BookInfoActivity extends Activity {

    protected void onStart() {
        ...
        setupCover(myBook);
        ...
    }

    private void setupCover(Book book) {
        final Bitmap coverBitmap = data.getBitmap(2 * maxWidth, 2 * maxHeight);
        if (coverBitmap == null) {
            return;
        }

        coverView.setVisibility(View.VISIBLE);
        coverView.getLayoutParams().width = maxWidth;
        coverView.getLayoutParams().height = maxHeight;
        coverView.setImageBitmap(coverBitmap);
    }
}
private class DecodeImageTask extends AsyncTask<Object, Void, Bitmap> {
    protected Long doInBackground(Object... args) {
        ZLAndroidImageData data = (ZLAndroidImageData) args[0];
        Integer maxWidth = (Integer) args[1];
        Integer maxHeight = (Integer) args[2];
        return data.getBitmap(2 * maxWidth, 2 * maxHeight);
    }

    protected void onProgressUpdate(Integer... progress) {
        // what’s to be done here?
    }

    protected void onPostExecute(Bitmap coverBitmap) {
        if (coverBitmap == null) {
            return;
        }
        coverView.setVisibility(View.VISIBLE);
        coverView.getLayoutParams().width = maxWidth;
        coverView.getLayoutParams().height = maxHeight;
        coverView.setImageBitmap(coverBitmap);
    }

    protected void onPostExecute() {}
}