The University Of Sheffield.

The Future Of Search-Based Software Testing?

Phil McMinn

including joint work with

Sheeva Afshan, Gordon Fraser, Mathew Hall, Muzammil Shahbaz & Mark Stevenson

“Where is Sheffield?”

5th largest city in the UK

Search-Based Testing

Software Testing

Execute a piece of software to gain confidence that it works correctly

However, finding bugs is difficult: the number of possible inputs to a program is enormous if not infinite!

Search-based software testing

Treat testing as a search-based optimisation problem – the *input space is the search space*

Phil McMinn – p.mcminn@sheffield.ac.uk

Automatic Test Input Generation has long been concerned simply with achieving

branch coverage

```
public boolean validateEmail(String text) {

    if (text.length() > 255)
        return false;
    int atPos = text.indexOf('@');
    if (atPos < 1 || atPos > 64 || atPos >= text.length() - 1)
        return false;
    if (text.charAt(atPos - 1) == '.'
            || text.charAt(text.length() - 1) == '.')
        return false;

    // Check local part
    boolean nextDotValid = false;
    for (int i = 0; i < atPos; ++i) {
        char c = text.charAt(i);
        boolean isOk = false;
        isOk = isOk || Character.isLetter(c);
        isOk = isOk || Character.isDigit(c);
        isOk = isOk || c == '!' || c == '#' || c == '$' || c == '%'
                || c == '&' || c == '*' || c == '+' || c == '-' || c == '/'
                || c == '=' || c == '?' || c == '^' || c == '_' || c == '`'
                || c == '{' || c == '|' || c == '}' || c == '~';
        if (c == '.') {
            isOk = nextDotValid;
```

Branch Coverage

Techniques like Dynamic Symbolic Execution have now developed to the point where they can easily obtain branch coverage (and other types of structual coverage)

So the question is …

… what next for search-based software testing?

With the power of search-based optimization, search-based testing is capable of a lot more …

Phil McMinn – p.mcminn@sheffield.ac.uk

Typically, generated test cases and the resulting outputs will have to be evaluated by a human in order to check for correctness

How to reduce human-checking effort?

How to reduce human oracle cost?

Reduce the *amount* of work

generate test data that maximises [illegible]ge

but minimises the

Quantitative approaches

[illegible] 5:166.

[illegible]tests generated in search based test data generation

[illegible]plication to the oracle cost problem. SBST 2010

Evolutionary algorithms for the multi−objective test data generation problem
J Ferrer, F Chicano, E Alba
Software: Practice and Experience

reduce size of test cases

A. Leitner, M. Oriol, A. Zeller, I. Ciupa, and B. Meyer. Efficient unit test case minimization.
ASE 2007, pp. 417–420. ACM.

How to reduce human oracle cost?

Reduce the *difficulty* of the work

how easily can the scenario comprising a
test case be understood
so it can be evaluated for correctness?

How to reduce human oracle cost?

Reduce the *difficulty* of the work

how easily can the scenario comprising a generated test inputs be understood so it can be evaluated for correctness?

Typical Automatic Test Case Generation

-4048
-10854
-29141
3140
733
....

Machine-generated test data tends to not fit the operational profile of a program particularly well

Typical Automatic Test Case Generation

!&^@s.sd

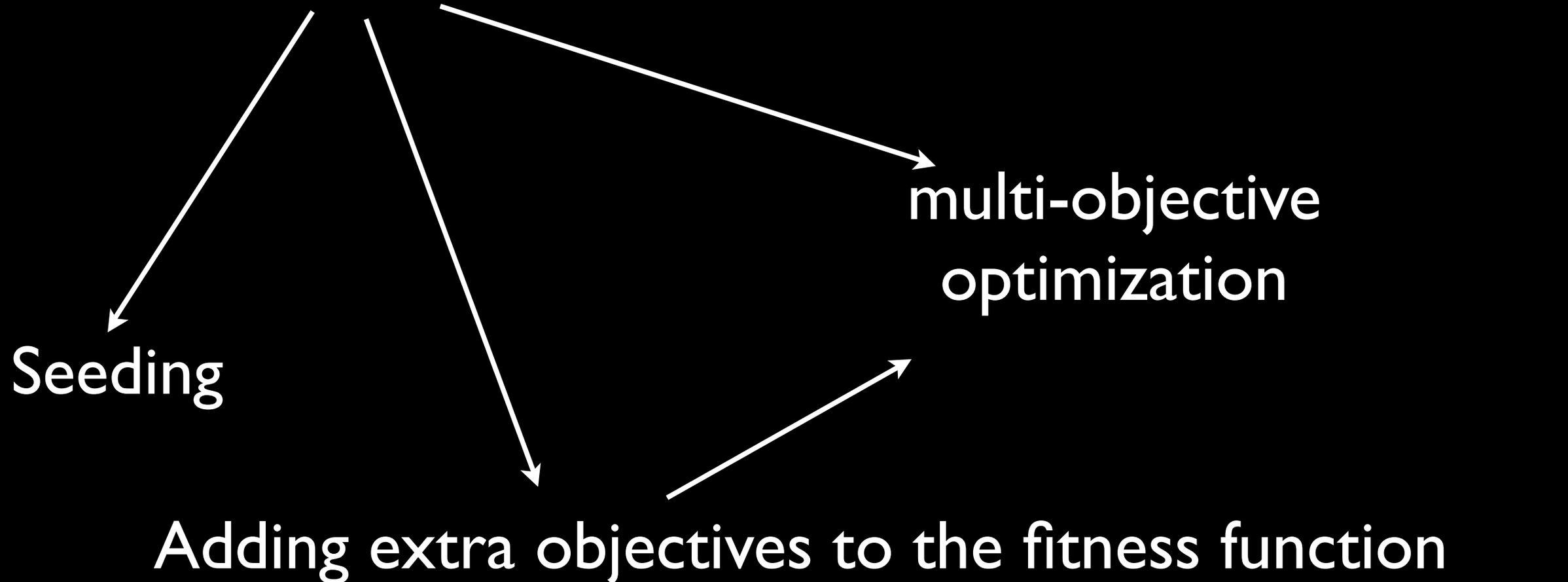
Valid

bill@microsoft.com

Readable, realistic

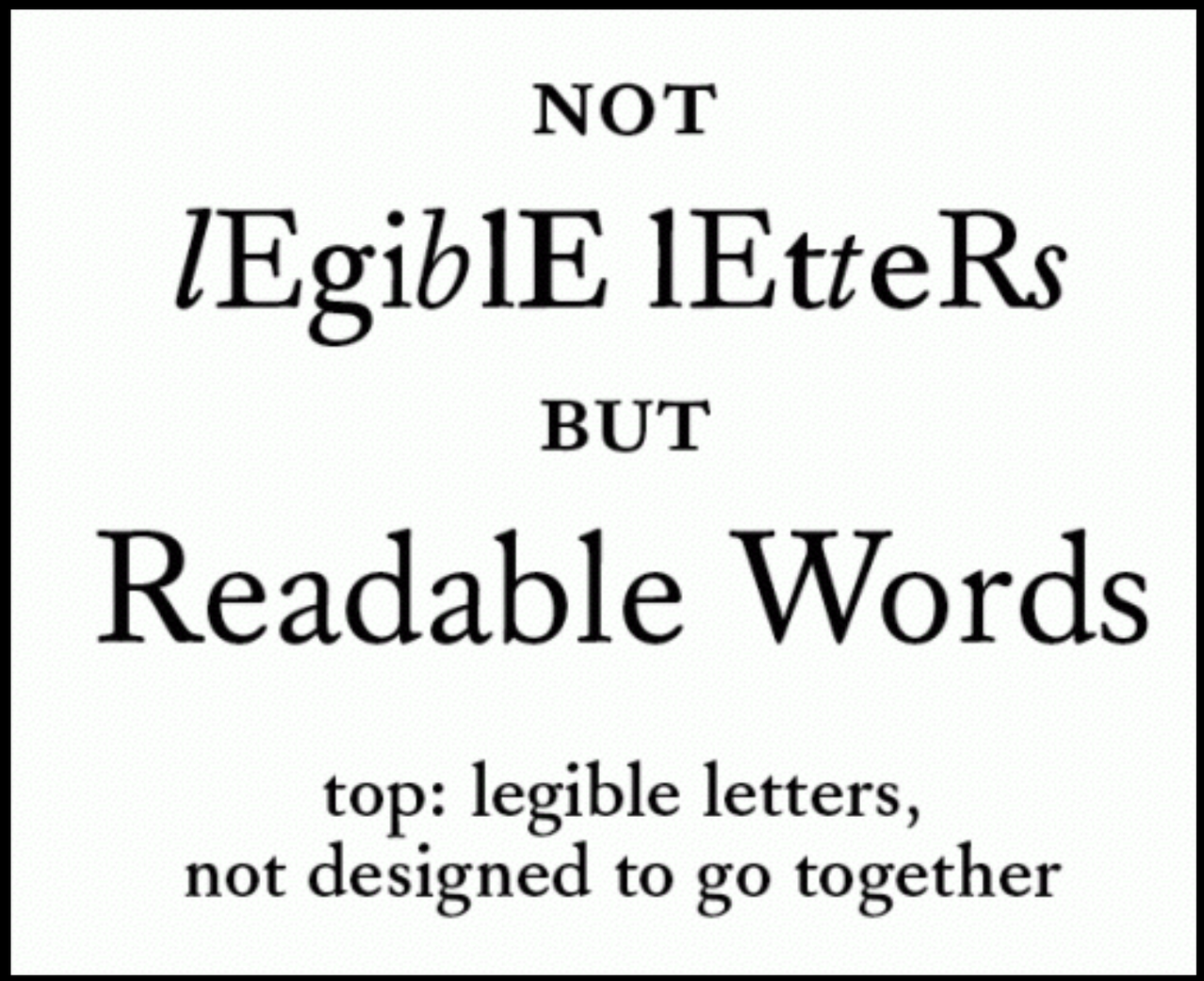
SBST to the rescue...

SBST to the rescue...



Incorporation of a Language Model for String Generation

Give better fitness values to strings with character combinations that occur naturally



S. Afshan, P. McMinn and M. Stevenson. "Evolving Readable String Test Inputs Using a Natural Language Model to Reduce Human Oracle Cost". Proc. ICST 2013

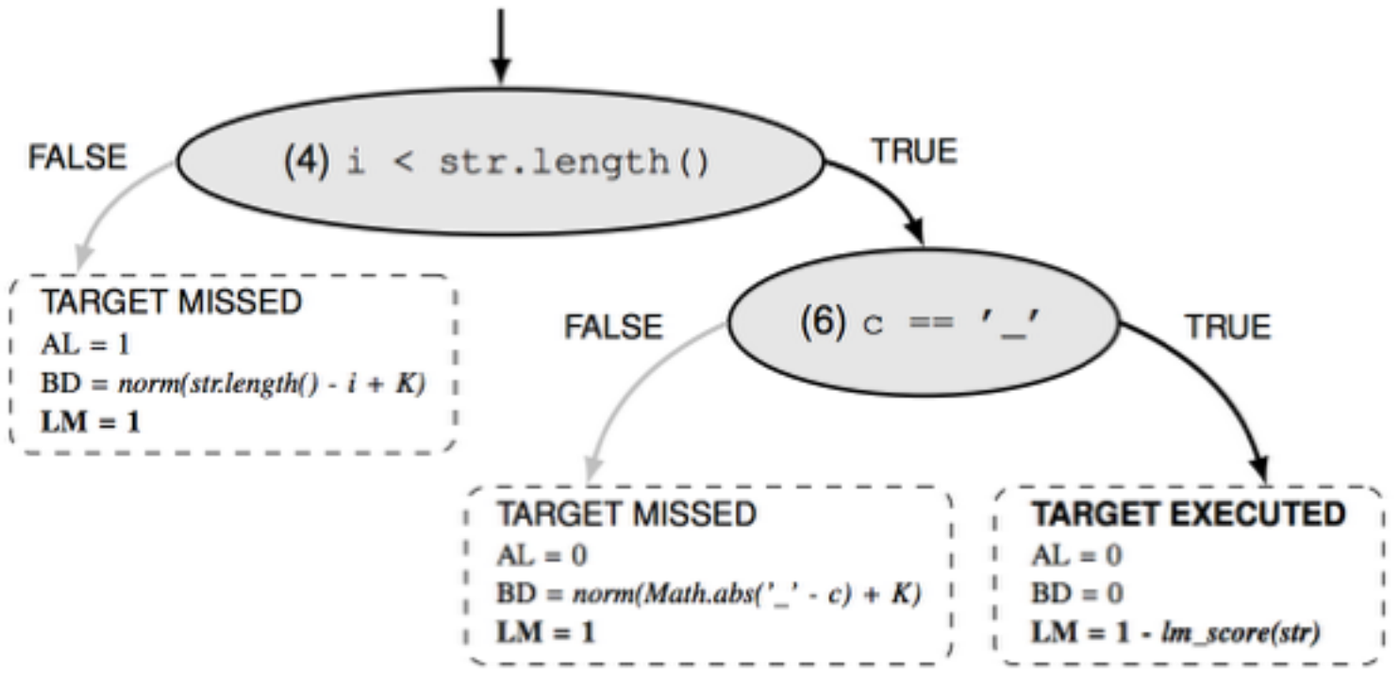
Bigram	Probability	Source
te	0.08548796	Direct
es	0.10209079	Direct
st	0.17185259	Direct
ti	0.07612985	Direct
in	0.30709963	Direct
ng	0.15313497	Direct

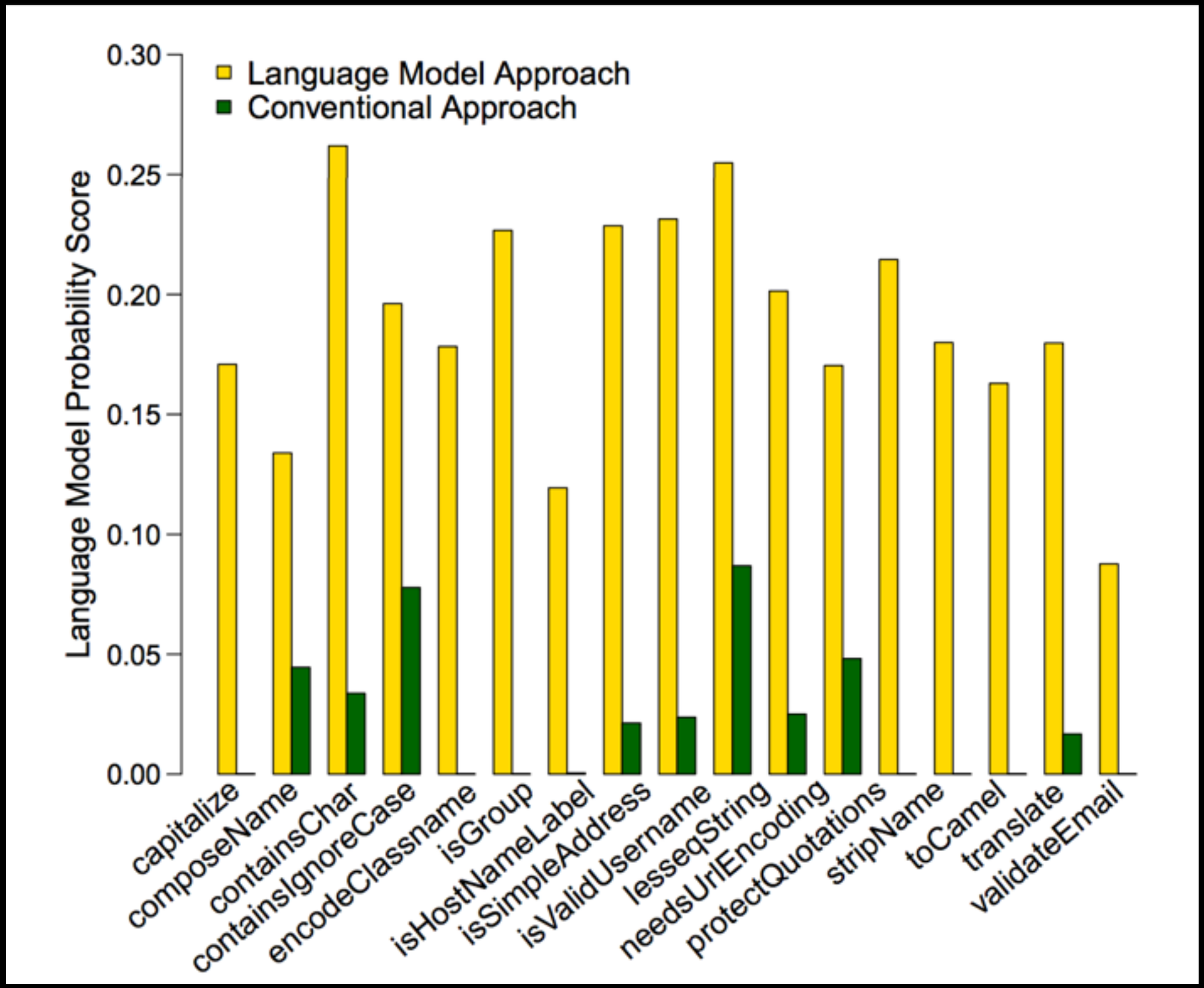
***score*(“*testing*”) = 0.17665935**

Bigram	Probability	Source
Qu	0.95987654	Direct
u5	0.00000009	Inferred
5$	0.00000005	Inferred
$-	0.00074450	Inferred
-u	0.00247280	Direct
ua	0.02245566	Direct

***score*(“*Qu5$-ua*”) = 0.00079785**

```
1   String toCamel(String str) {
2     StringBuffer sb = new StringBuffer();
3     boolean wasUnderline = false;
4     for (int i = 0; i < str.length(); i++) {
5       char c = str.charAt(i);
6       if (c == '_') {
7         wasUnderline = true;
8         continue;
9       }
10      if (wasUnderline) {
11        sb.append(Character.toUpperCase(c));
12        wasUnderline = false;
13        continue;
14      }
15      sb.append(Character.toLowerCase(c));
16    }
17    return sb.toString();
18  }
```




Language Model Approach
Conventional Approach
Language Model Probability Score
0.30
0.25
0.20
0.15
0.10
0.05
0.00
capitalize
composeName
containsChar
containsIgnoreCase
encodeClassname
isGroup
isHostNameLabel
isSimpleAddress
isValidUsername
lesseqString
needsUrlEncoding
protectQuotations
stripName
toCamel
translate
validateEmail

Human Study

protectQuotations

The following method takes a string argument, and places a backslash (\) in front of each quotation mark ("). The return value of this method is the string argument with a backslash placed in front of every occurring quotation mark.

```
String protectQuotations(String text){
  ....
}
```

Click next to answer 8 questions about this method.

The output produced by this method for the string input **"Nout** is: Skip This Question

Save and Proceed

Improvements in Correct Judgments

Case Study	Lang. (%)	Conv. (%)	*p*-value
capitalize	74.4	79.6	0.635
composeName	80.4	82.0	0.894
containsChar	94.0	90.0	0.747
containsIgnoreCase	85.6	84.8	0.948
encodeClassname	**97.2**	**74.8**	**0.048**
getClassName	83.6	80.4	0.790
isGroup	95.6	96.8	0.949
isHostNameLabel	87.6	86.4	0.948
isSimpleAddress	94.4	90.4	0.747
isValidUsername	87.2	94.0	0.604
lesseqString	78.4	78.0	1.000
needsUrlEncoding	95.6	96.8	0.949
protectQuotations	88.0	84.8	0.793
stripName	**90.0**	**59.2**	**0.003**
toCamel	**90.4**	**59.2**	**0.003**
translate	88.4	84.8	0.793
validateEmail	72.0	89.6	0.108

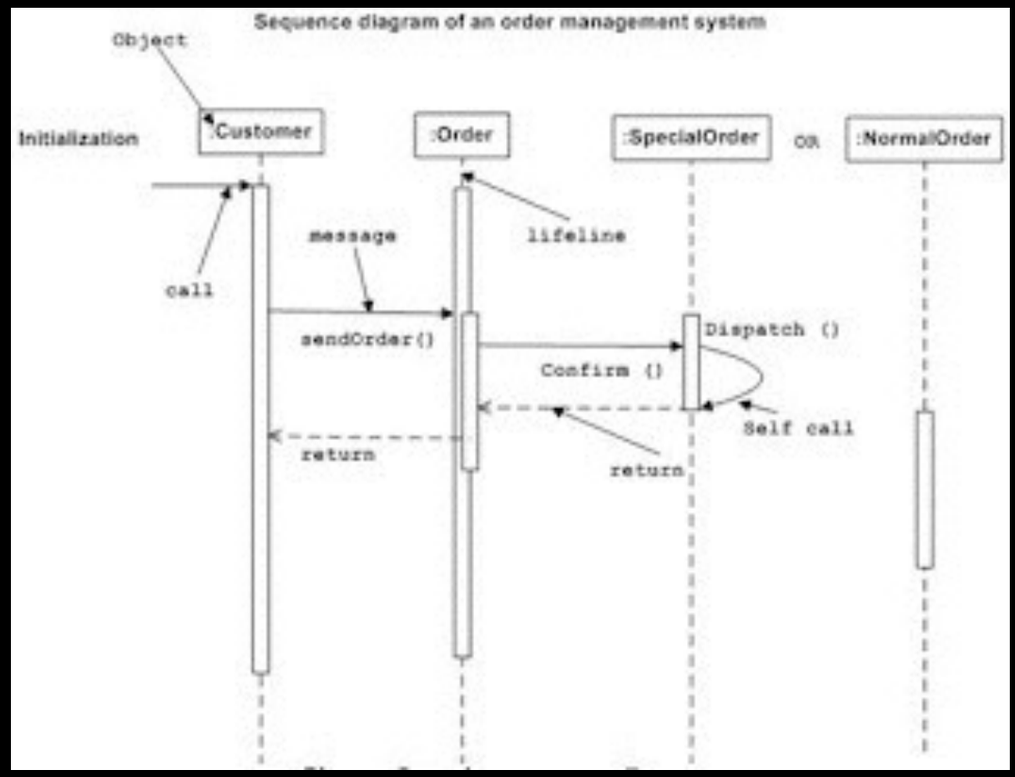
Improvements in Assessment Time

Case Study	Lang. (s)	Conv. (s)	p-value	$\hat{A}_{12}$
capitalize	14.1	16.6	0.820	0.506
composeName	20.6	21.7	0.596	0.514
containsChar	**9.1**	**14.3**	**< 0.001**	*** **0.279**
containsIgnoreCase	10.4	10.2	0.877	0.496
encodeClassname	**15.1**	**40.3**	**< 0.001**	*** **0.100**
getClassName	**10.5**	**19.5**	**< 0.001**	** **0.328**
isGroup	**6.1**	**6.8**	**0.006**	* **0.429**
isHostNameLabel	**6.9**	**8.9**	**< 0.001**	* **0.414**
isSimpleAddress	**7.9**	**13.9**	**< 0.001**	** **0.350**
isValidUsername	6.2	5.5	0.946	0.502
lesseqString	**14.0**	**24.0**	**< 0.001**	* **0.407**
needsUrlEncoding	8.6	8.8	0.114	0.541
protectQuotations	**13.1**	**15.8**	**< 0.001**	* **0.380**
stripName	**20.4**	**42.7**	**< 0.001**	*** **0.194**
toCamel	**16.4**	**33.6**	**< 0.001**	*** **0.193**
translate	15.5	25.6	0.089	0.456
validateEmail	9.8	7.5	0.136	0.539

Extra Objectives

Common call sequence patterns

Gordon Fraser, Andreas Zeller.
Exploiting Common Object Usage in
Test Case Generation. Proc. ICST 2011



Give better fitness values to test cases with method call sub-sequences that occur in practice

Seeding

Programmer sanity checks

P. McMinn, M. Stevenson and M. Harman. "Reducing Qualitative Human Oracle Costs associated with Automatically Generated Test Data". Proc. STOV 2010

Already-existing tests

Gordon Fraser, Andrea Arcuri: The Seed is Strong: Seeding Strategies in Search-Based Software Testing. Proc. ICST 2012

Other sources

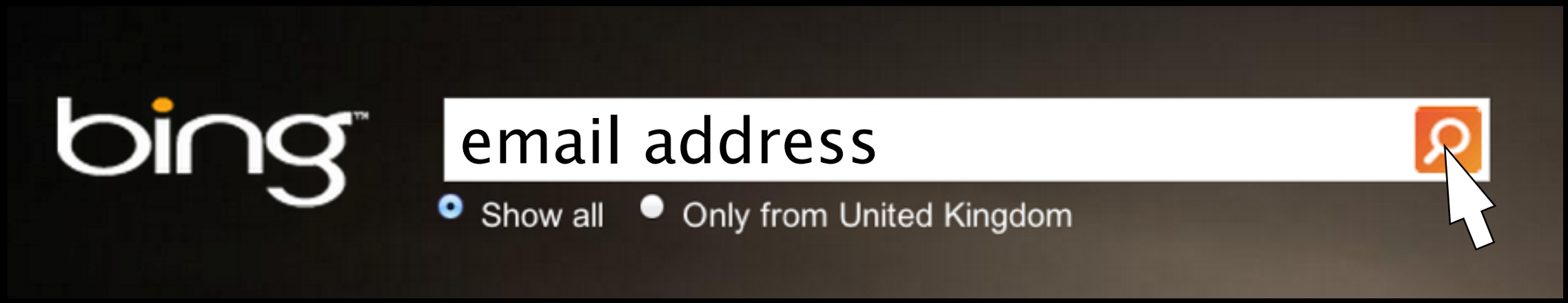
Documentation

Project wikis

Actual usage of the software

Email/IM conversations?

Using the Internet for String examples



WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact Wikipedia

Toolbox

Print/export

Languages
Česky
Dansk

Email address

From Wikipedia, the free encyclopedia

An **email address** identifies an email box to which email messages are delivered. An example format of an email address is `lewis@example.net` which is read as *lewis at example dot net*. Many earlier email systems used different address formats.

Contents [hide]

1 Overview
2 Syntax
2.1 Local part
2.2 Domain part
2.3 Examples
2.3.1 Valid email addresses
2.3.2 Invalid email addresses
3 Common local-part semantics
3.1 Local-part normalization
3.2 Address tags
4 Validation
4.1 Identity validation
5 Internationalization
5.1 Internationalization Examples
5.2 Internationalization Support
6 See also
7 References

Email address

From Wikipedia, the free encyclopedia

An **email address** identifies an email box to which email messages are delivered. An example format of an email address is `lewis@example.net` which is read as *lewis at example dot net*. Many earlier email systems used different address formats.

Email addresses, such as `jsmith@example.org` have two parts. The part sign is the *local-part* of the address, often the username of the recipient (`jsmi` part after the @ sign is a *domain name* to which the email message will be se (`example.org`).

An **email address** identifies an email box to which email messages are delivered. An example format of an email address is `lewis@example.net` which is read as *lewis at example dot net*. Many earlier email systems used different address formats.

Email addresses, such as `jsmith@example.org` have two parts. The part sign is the *local-part* of the address, often the username of the recipient (`jsmi` part after the @ sign is a *domain name* to which the email message will be se (`example.org`).

P. McMinn, M. Shahbaz and M. Stevenson.
"Search-Based Test Input Generation for String Data Types Using the Results of Web Queries".
Proc. ICST 2012

Phil McMinn – p.mcminn@sheffield.ac.uk

Knowing what to look for

```
class Mailer {

    boolean isValid(String emailAddress) {

        // ...
    }
}
```

Knowing what to look for

```
class Util {

    boolean isEmailAddress(String str) {

        // ...
    }
}
```

Knowing what to look for

```
class EmailAddress {

    boolean isValid(String str) {

        // ...
    }
}
```

Phil McMinn – p.mcminn@sheffield.ac.uk

From identifiers to queries

an_email_address

Phil McMinn – p.mcminn@sheffield.ac.uk

Remove stop words

an email address

base query

Phil McMinn – p.mcminn@sheffield.ac.uk

Query Generation

	Pluralised	Quoted	Prefixed
email address (base query)	✗	✗	✗
email addresses	✔	✗	✗
“email addresses”	✔	✔	✗
“list of email addresses”	✔	✔	✔

Phil McMinn – p.mcminn@sheffield.ac.uk

Process

1. Extract identifiers relevant to each string method parameter

2. Construct base queries (split identifiers into words)

3. Generate set of queries

4. Perform queries on Bing, download resulting web pages

5. Tokenise web pages according to white space

6. Store unique strings found in a list

Phil McMinn – p.mcminn@sheffield.ac.uk

Process

1. Extract identifiers relevant to each string method parameter

2. Construct base queries (split identifiers into words)

3. Generate set of queries

4. Perform queries on Bing, download resulting web pages

5. Tokenise web pages according to white space

6. Store unique strings found in a list

Phil McMinn – p.mcminn@sheffield.ac.uk

Process

1. Extract identifiers relevant to each string method parameter

2. Construct base queries (split identifiers into words)

3. Generate set of queries

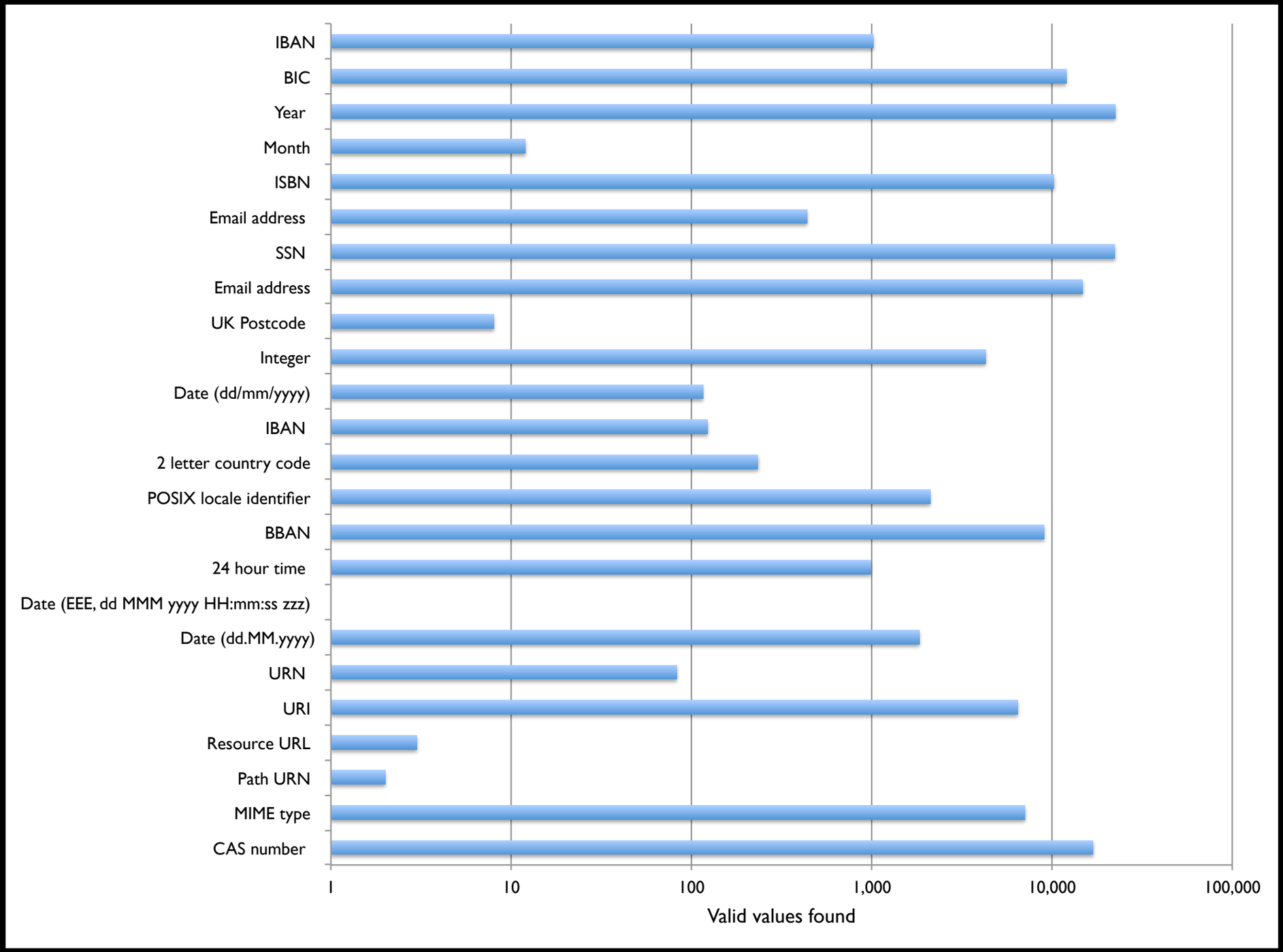
4. Perform queries on Bing, download resulting web pages

5. Tokenise web pages according to white space

6. Store unique strings found in a list

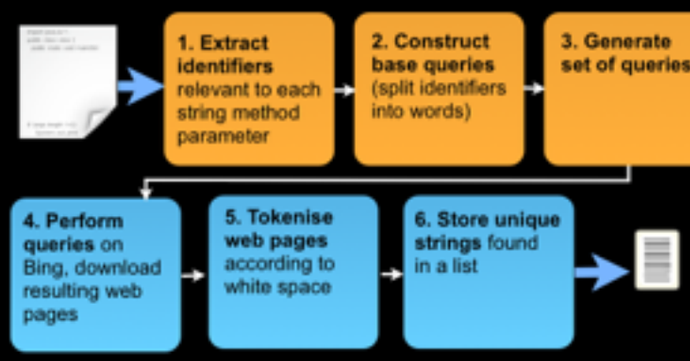
Phil McMinn – p.mcminn@sheffield.ac.uk

Testing String Validation Routines



Phil McMinn – p.mcminn@sheffield.ac.uk

String input generation



Phase 1.
Initial Evolutionary Searches

```
DDMMYYYYDate d = new DDMMYYYYDate();
d.parse("1//");
d.isValid();
```

Phase 2.
Injection of Web Values

```
DDMMYYYYDate d = new DDMMYYYYDate();
d.parse("18/4/2012");
d.isValid();
```

Phase 3.
Further Evolutionary Searches

```
DDMMYYYYDate d = new DDMMYYYYDate();
d.parse('18/47/2012")
d.isValid();
```

Phil McMinn – p.mcminn@sheffield.ac.uk

Result

- Readable test cases
- Improvement in coverage!
  - 75% of programs analysed experienced an average increase in coverage of 14%

Phil McMinn – p.mcminn@sheffield.ac.uk

Other Ideas for Qualitative Reduction

- Model programmers and their tests
- Readable/understandable outputs
- Generate inputs that more obviously fall into a certain partition of the input/output space such as valid or invalid

Comments to improve test case understanding

Current

```
public void testExe4()  throws Throwable  {

    Triangle triangle0 = new Triangle();
    int int0 = 249;
    int int1 = 911;
    int int2 = 911;

    int int3 = triangle0.exe(int0, int1, int2);

    assertEquals(3, int3);
}
```

Comments to improve test case understanding

Ideal

```
public void testIsoceles() {
    Triangle triangle = new Triangle();
    int a = 249;
    int b = 911;
    int c = 911;

    // test for an isoceles triangle
    int type = triangle.exe(a, b, c);

    assertEquals(3, type);
}
```

Comments to improve test case understanding

Possible

(using Symbolic Execution)

```
public void testExe4()  throws Throwable  {

    Triangle triangle = new Triangle();
    int a = 249;
    int b = 911;
    int c = 911;

    // simplified path constraints: c < (a + b) and a < b and b == c

    // This test is similar to the test 'testExe0' except a < b

    // This is the only test that exercises the following condition(s): a < b and b == c

    int type = triangle.exe(a, b, c);

    assertEquals(3, int3);
}
```

Fault-finding capability

Open question:

Does lowering human oracle cost hinder fault finding capability?

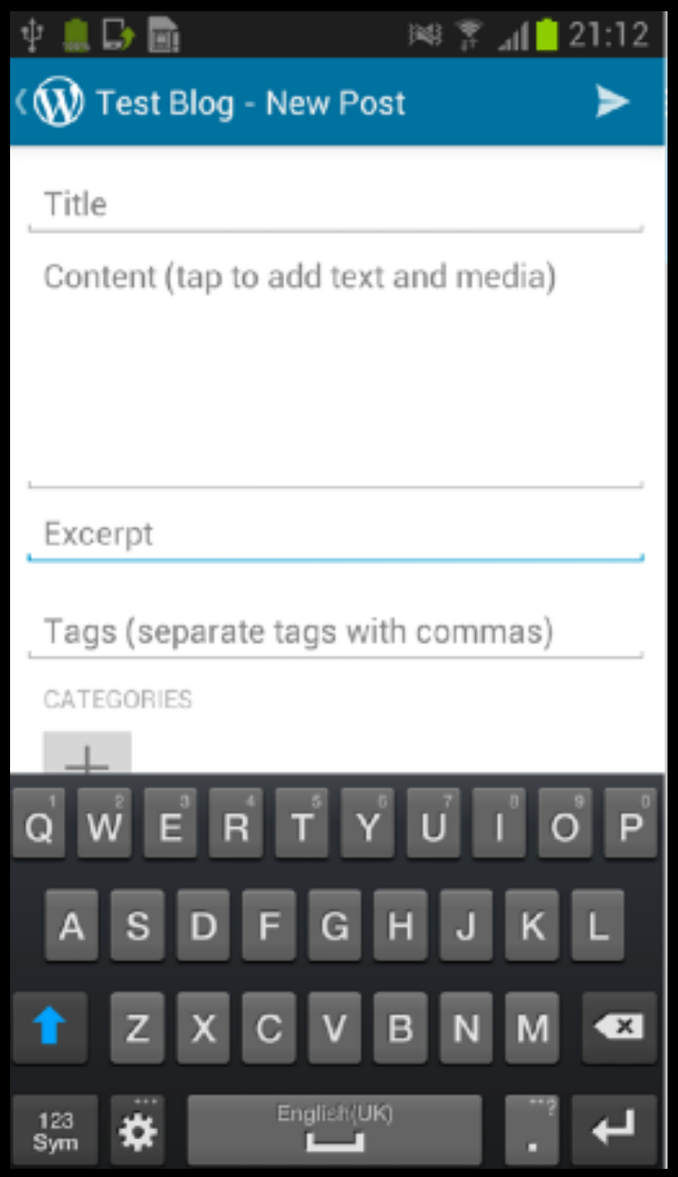
Non-Functional SBST

Because SBST is capable of generating test cases for non-functional aspects of software too…

Can Search-Based Testing Automate Energy Testing?

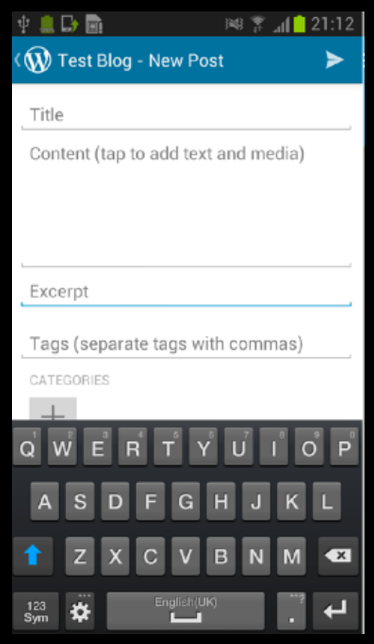
Some Initial Experiments

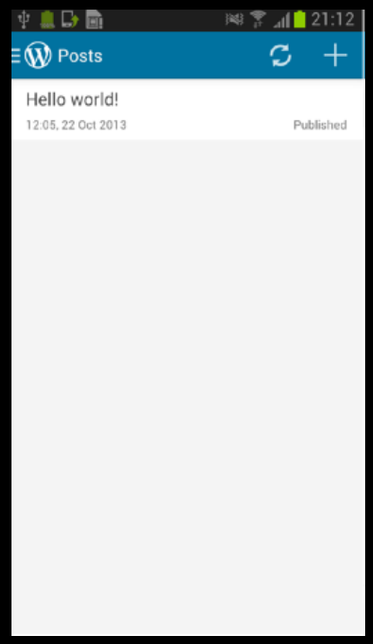
Testing Energy Consumption in an Android App



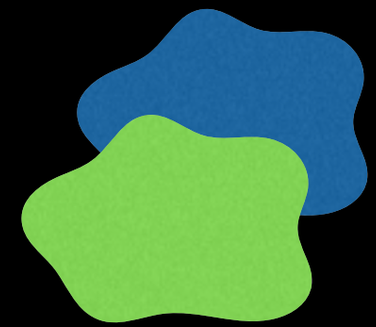
WordPress App

WordPress App



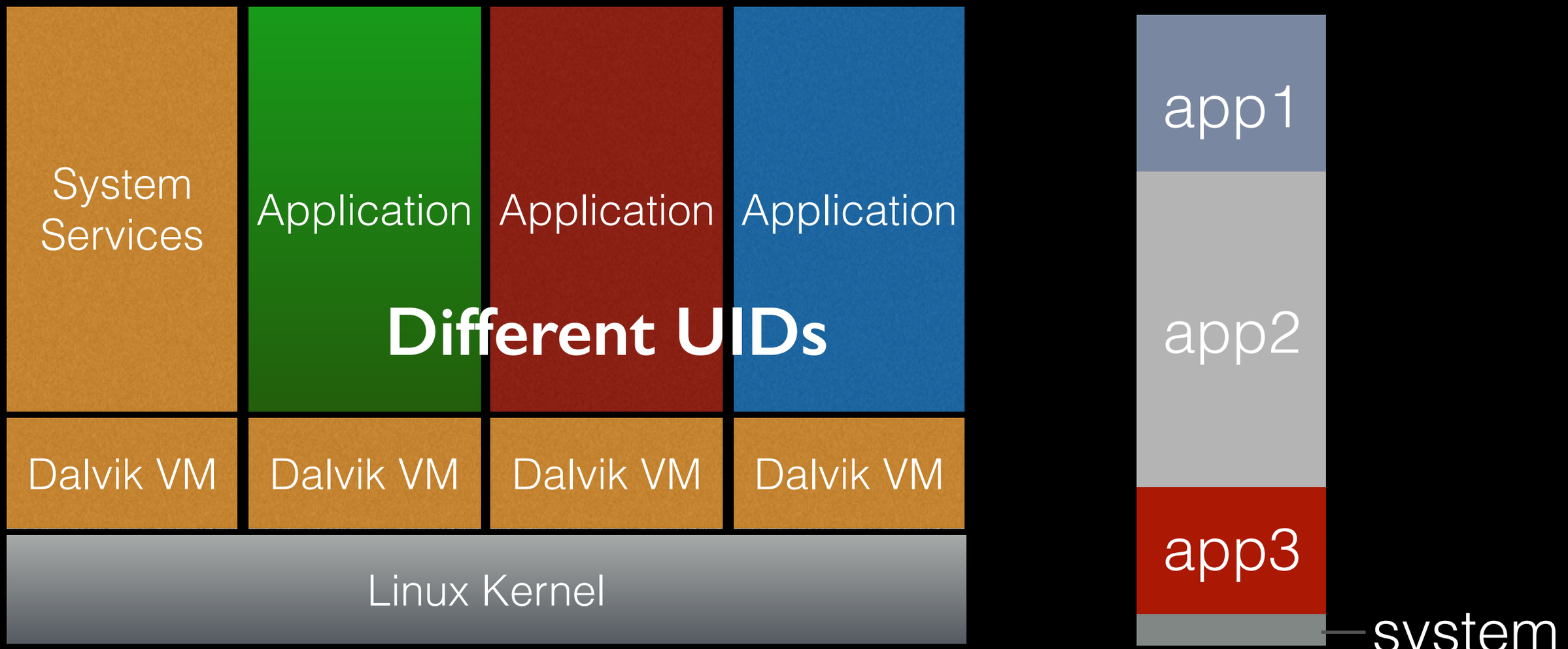


Activities



Services

App Isolation



CPU

GPU

Disk I/O

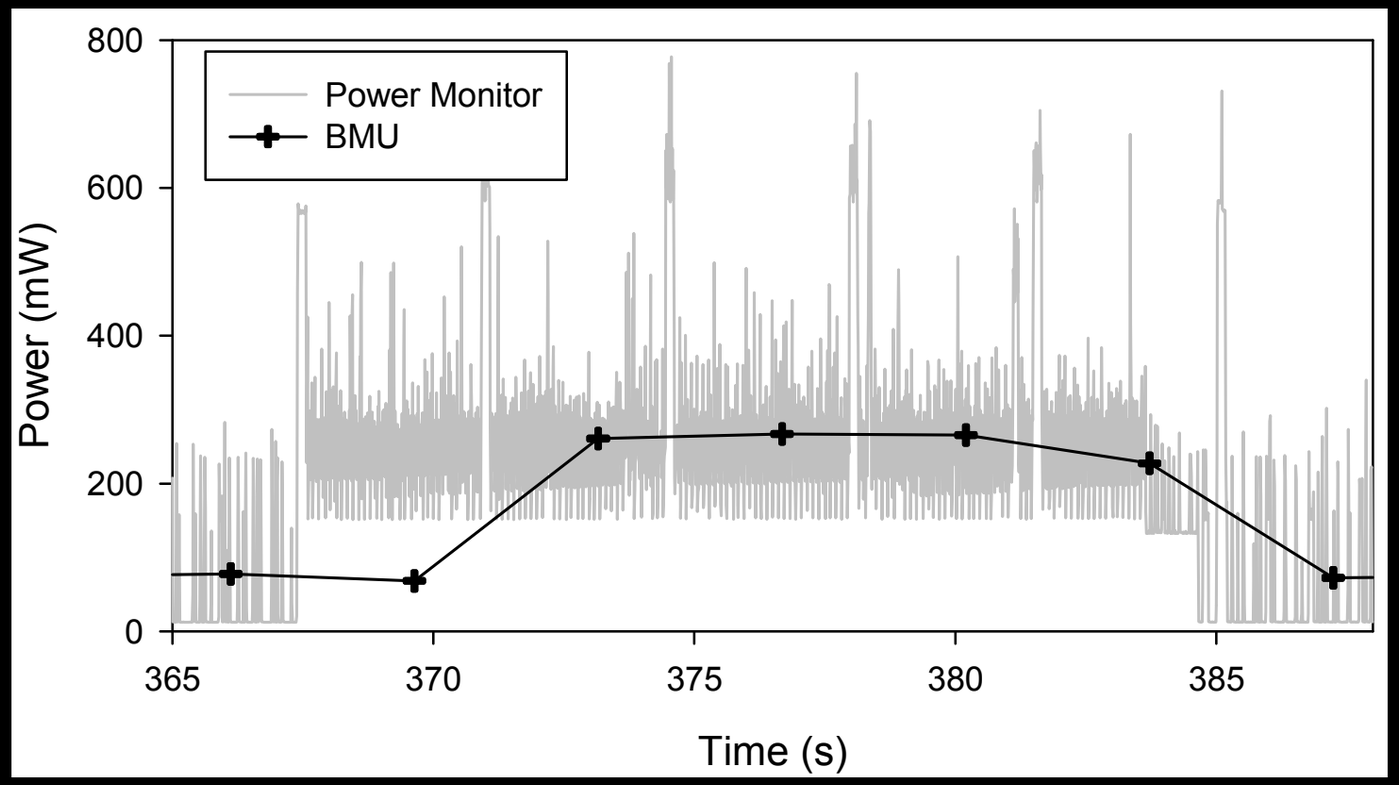
Camera

WiFi

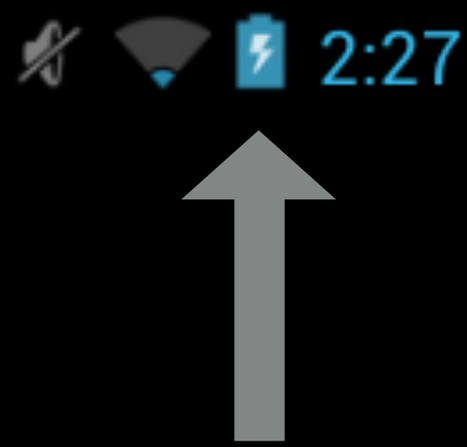
Mobile Network

GPS

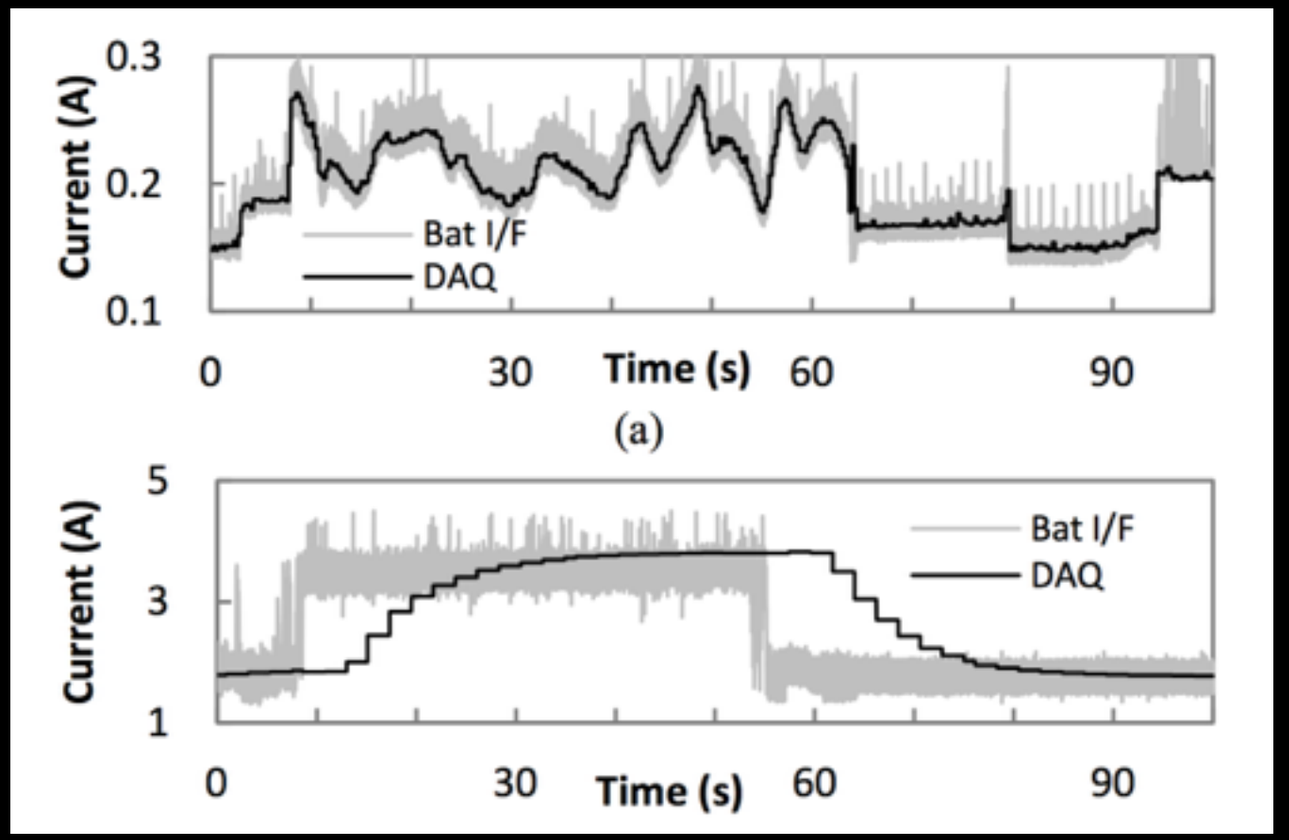
Sensors



Jung et al. 2012



“fuel gauge”



Dong and Zhong 2010

CPU

GPU

Disk I/O

Camera

WiFi

Mobile Network

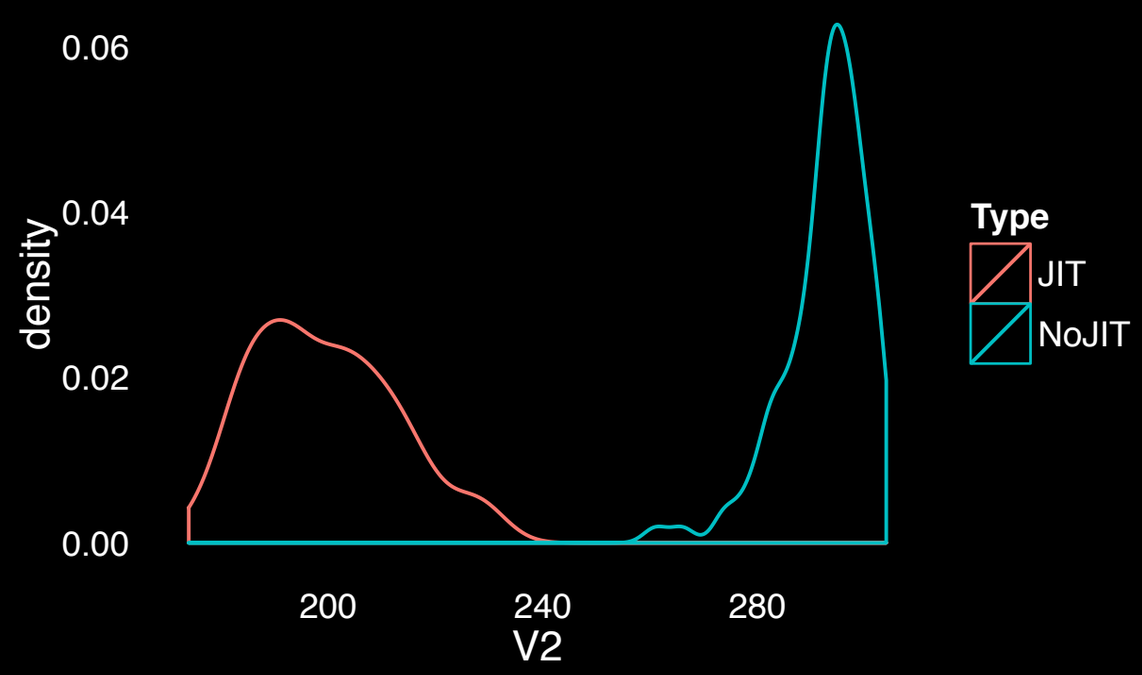
GPS

Sensors

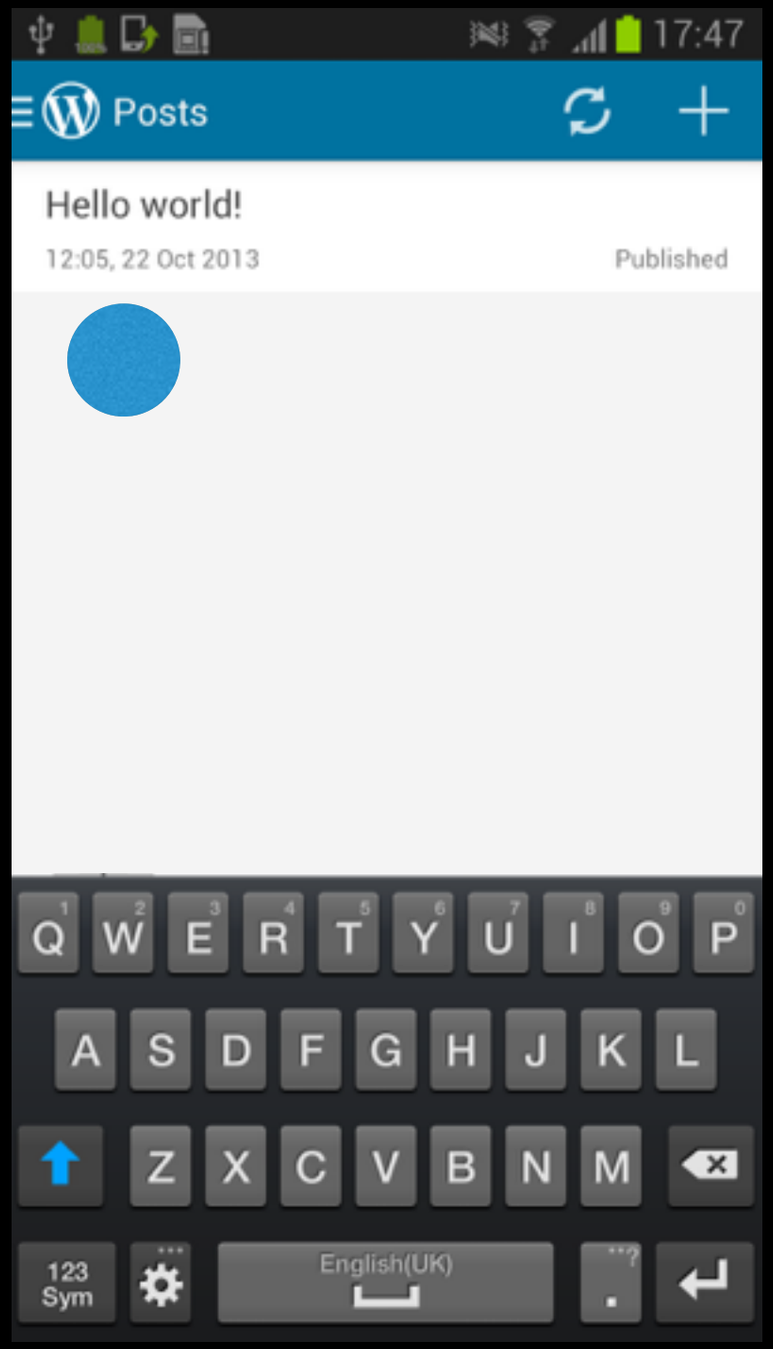
Estimated CPU Energy Consumption

/proc/pid/stat

- CPU energy is (approximately) linear with CPU time for a fixed frequency
- Nondeterminism: IO, ~~L1/L2 cache~~, ~~JIT~~, ~~Multicore~~



events

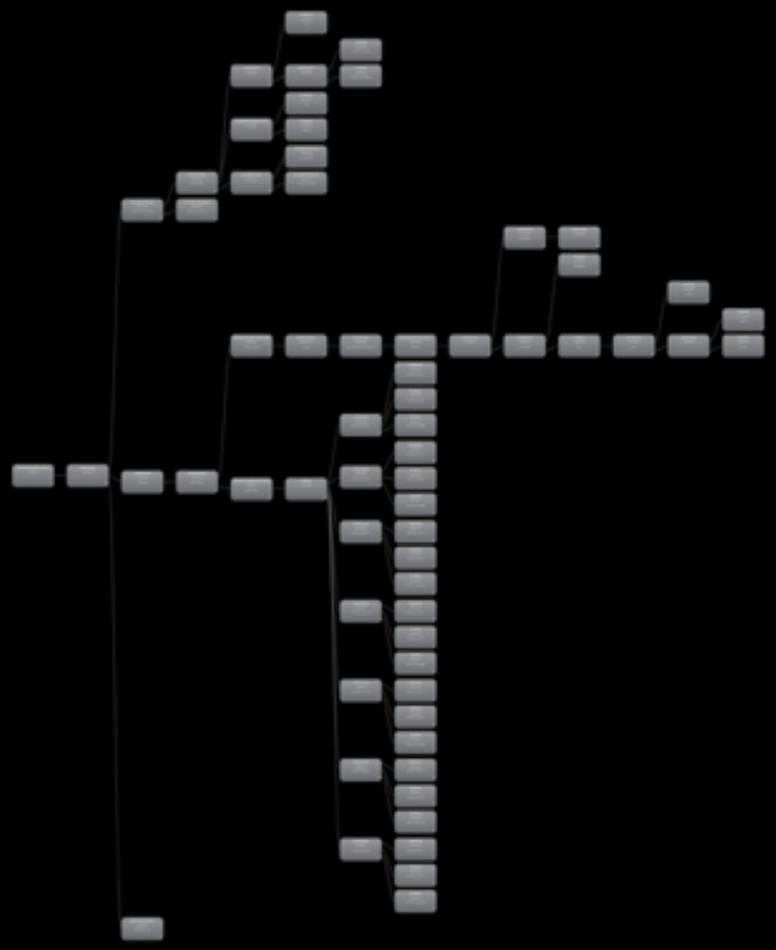


CPU time

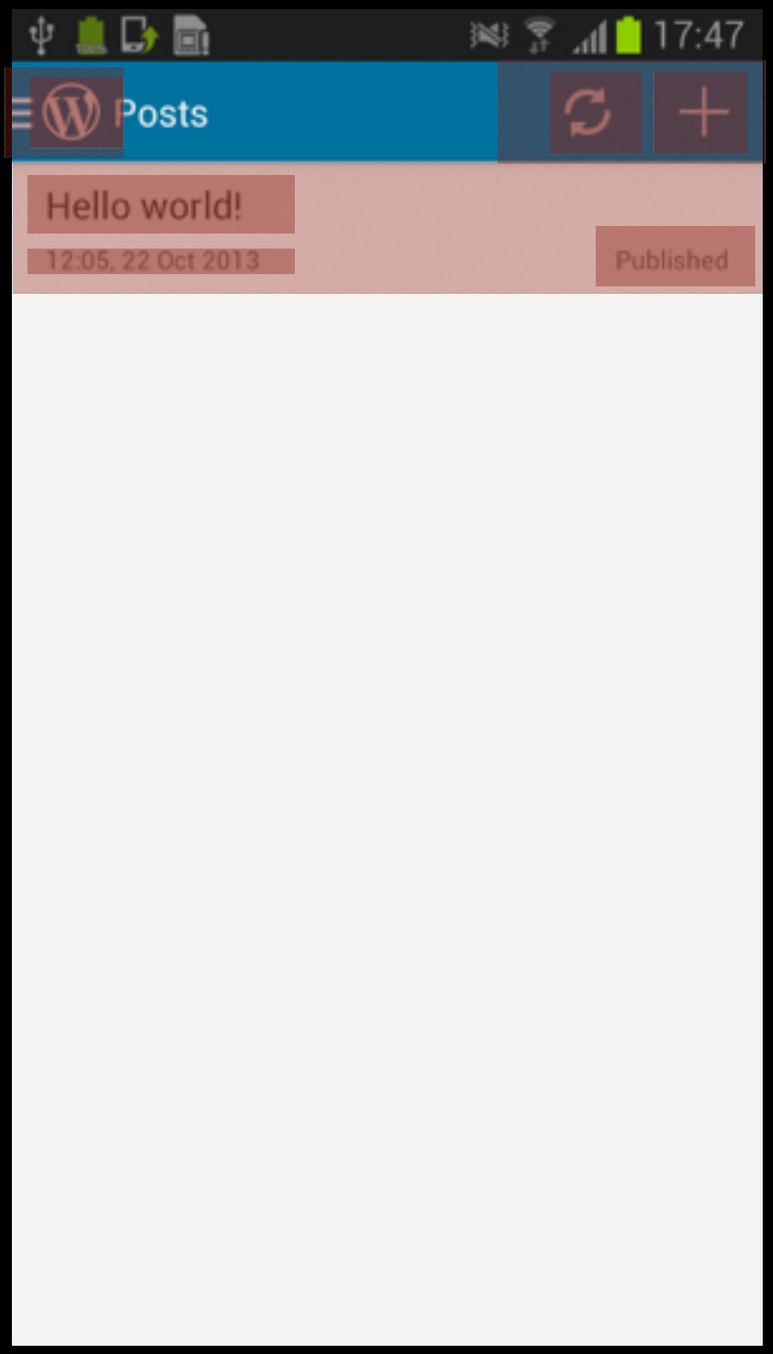
Estimated CPU Energy Consumption

events


17:47
Hello world!
12:05, 22 Oct 2013
Published
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M
123 Sym
English(UK)



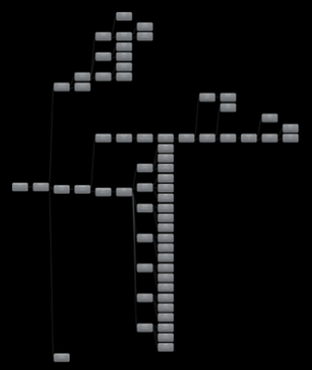
View State Graph



Robotium

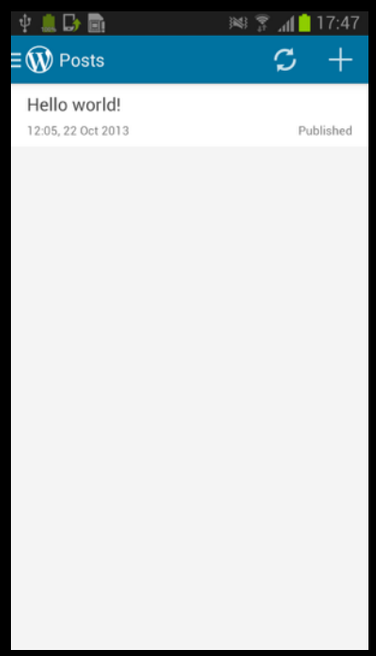
Process

View State Graph



Search

Perform Events



Estimated CPU Energy Consumption

(/proc/pid/stat)

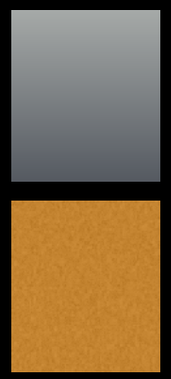
Search Process

- 15 UI Events (Click View, Enter Text in View, Back button)
- CPU time only (fixed frequency, no JIT)

Sequence construction

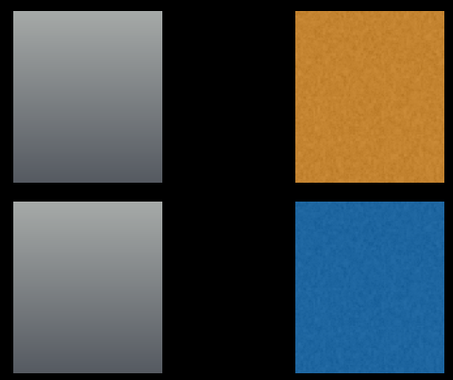
Sequence construction

Sequence construction

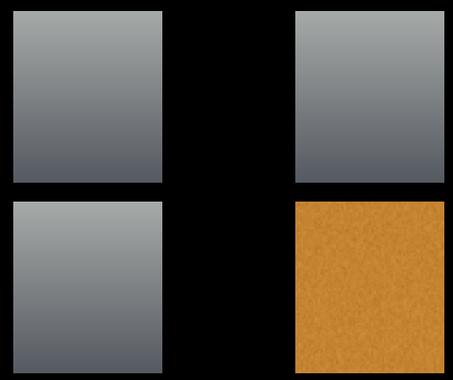


Sequence construction

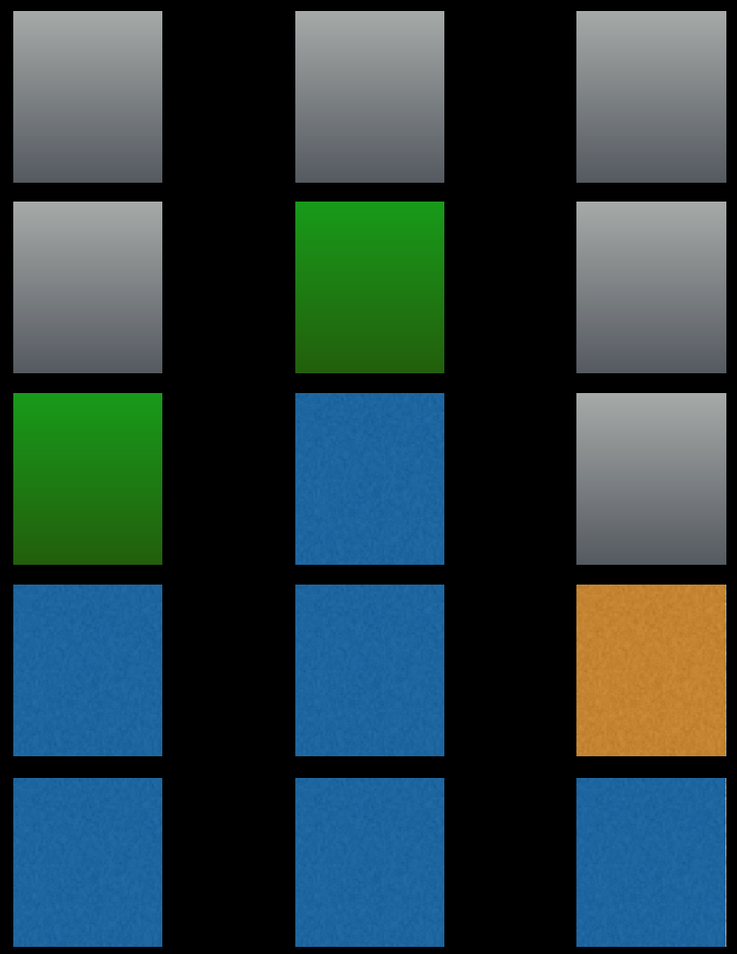
Sequence construction



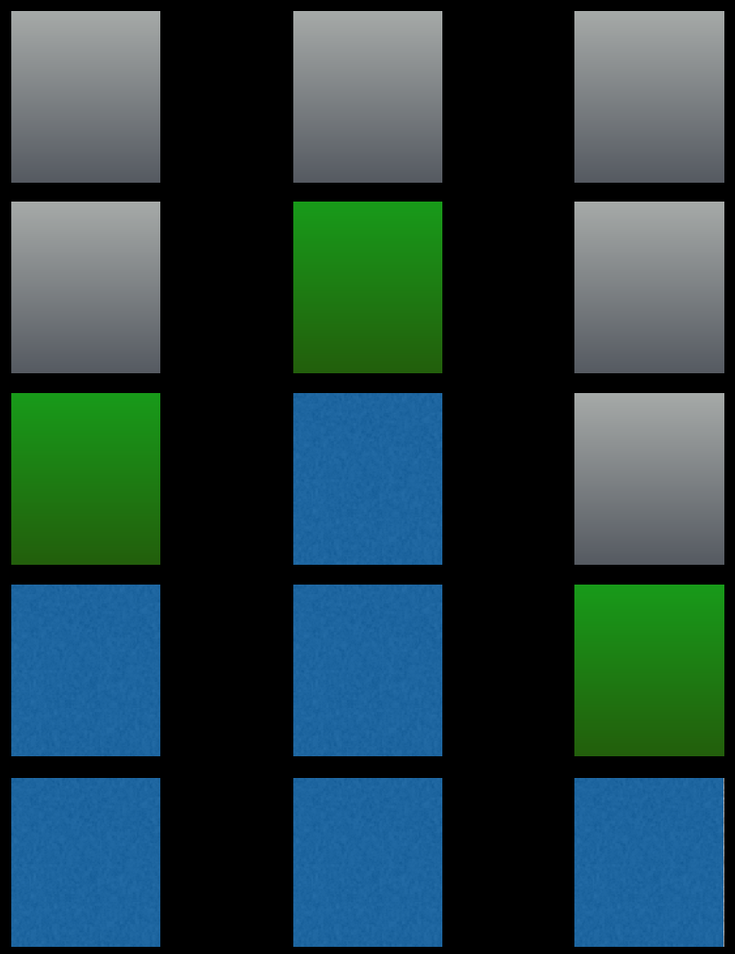
Sequence construction



Sequence construction



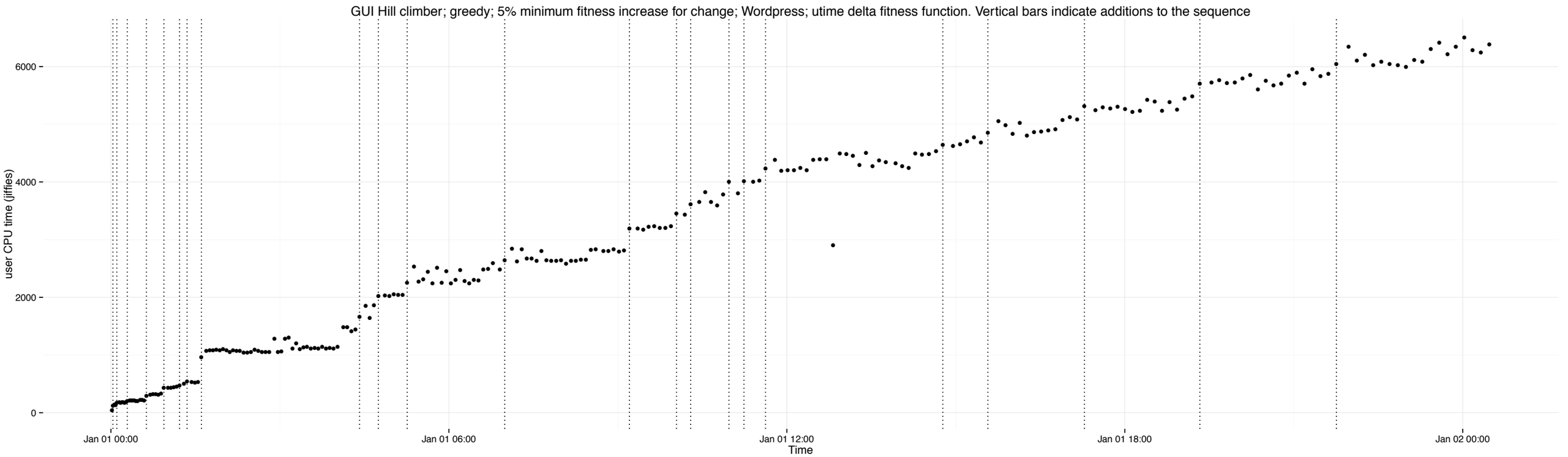
Sequence construction



Restart for each element to be enumerated

Time to enumerate inputs increases with sequence length

Result


GUI Hill climber; greedy; 5% minimum fitness increase for change; Wordpress; utime delta fitness function. Vertical bars indicate additions to the sequence
6000
4000
2000
0
user CPU time (jiffies)
Jan 01 00:00
Jan 01 06:00
Jan 01 12:00
Jan 01 18:00
Jan 02 00:00
Time

Conclusions of Expt

- Guided search can be used to try to find sequences that use the most power

- But …

Problems/Open Questions

What is a bug / anomalous power behaviour

- Too much usage?
- Unusual consumption?
    - How to characterise this

Problems/Open Questions

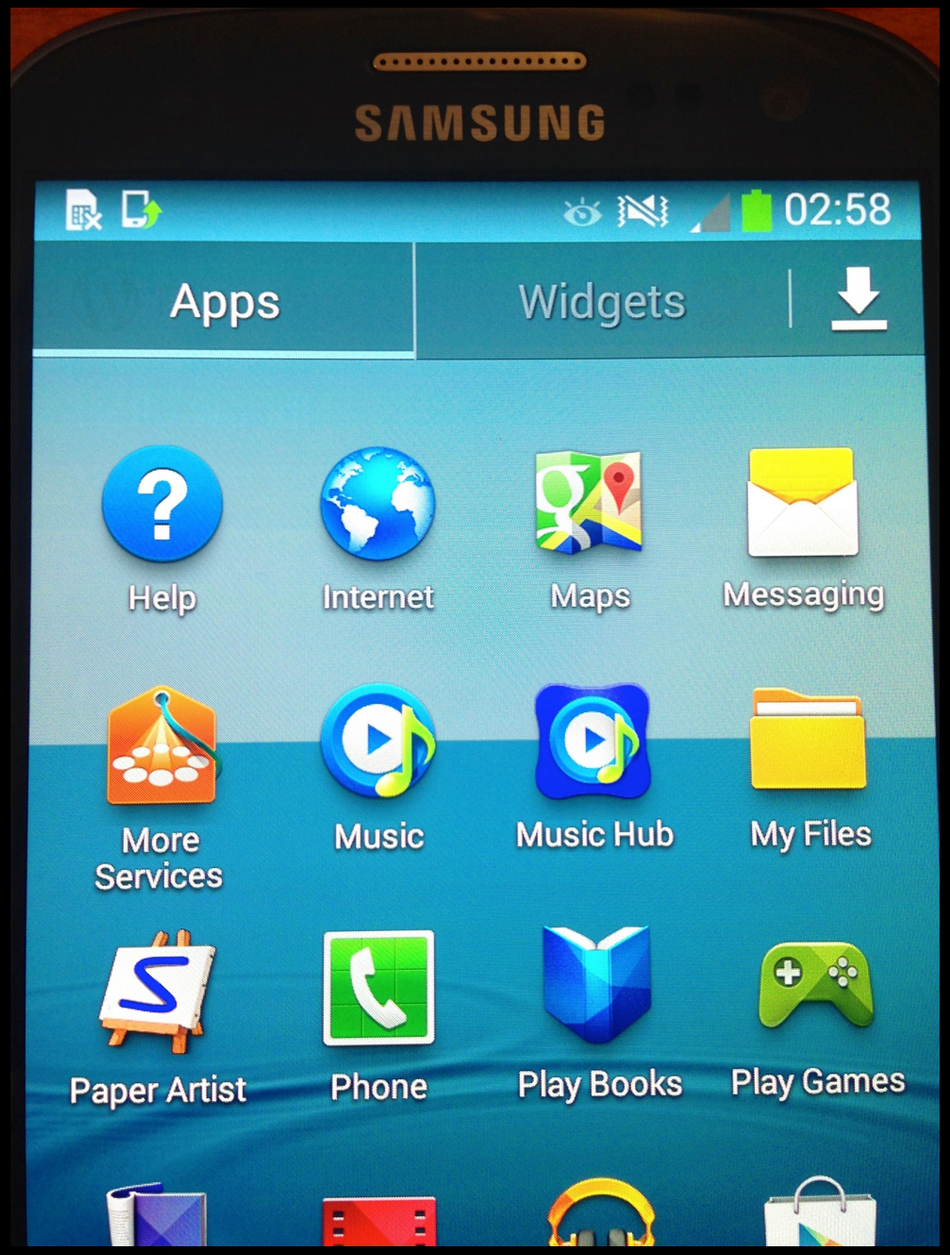
Measuring Energy

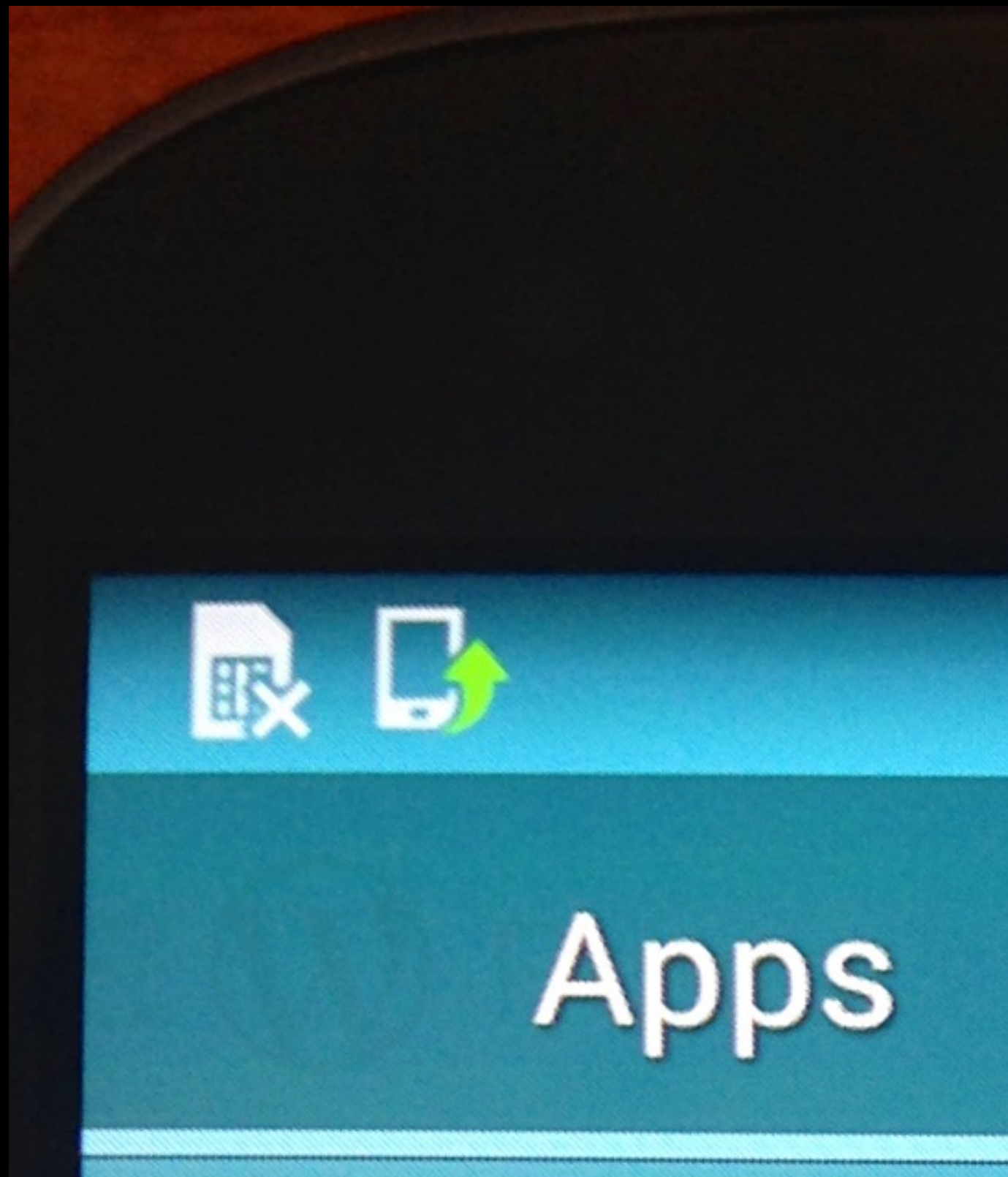
- Varying CPU/GPU frequency
- Caching
- Just in time compilation behaviour
- I/O and other services
- Attributing energy consumption back to an app

Problems/Open Questions

Performing Tests

- Very time consuming!
    - Repetitions needed to smooth chance effects
- No possibility to farm out to a traditional cluster/grid
    - (Unless we use a model/emulator, in which case our results are approximates)


SAMSUNG
02:58
Apps
Widgets
Help
Internet
Maps
Messaging
More Services
Music
Music Hub
My Files
Paper Artist
Phone
Play Books
Play Games


Apps

Problems/Open Questions

What level to apply testing?

- App level?
- Energy unit tests?

Can Search-Based Testing Automate Energy Testing?

- Encouraging initial results, but need…
  - better idea of where to target testing
  - reliable ways to measure energy
  - easy ways to perform and repeat expts.

Summary

- Search-based testing is more than just generating test cases for structural coverage of programs
- Two examples include:
    - The human oracle problem – generating readable test cases
    - Non-functional aspects such as testing for battery usage