Future Trends in Software Engineering

Wolfram Schulte
Software Engineering (SE)

“Produce high quality software with a finite amount of resources to a predicted schedule”

NATO SE Conference 1968
Agenda

Analytics for software development
  • *Many known programs: Branch analysis*

Logic-based tools
  • *Two known programs: Equivalence checking*

Future platforms, future developers
  • *One known, one unknown pgm: Coding duels*
Analytics

“Use of analysis, data, and systematic reasoning to make decisions”

- Financial services
- Manufacturing
- Health care
- Search
- And more…
### Analytics

<table>
<thead>
<tr>
<th>Past</th>
<th>Present</th>
<th>Future</th>
</tr>
</thead>
<tbody>
<tr>
<td>What happened? (Reporting)</td>
<td>What is happening now? (Alerting)</td>
<td>What will happen? (Extrapolation)</td>
</tr>
<tr>
<td>How and why did it happen (Modeling)</td>
<td>What’s the best next action? (Recommendation)</td>
<td>What’s the best/worst that can happen? (Prediction)</td>
</tr>
</tbody>
</table>

*From Davenport et al. “Analytics at Work”.*
Branching in Source Control Management (SCM) Systems

Coordinating the work of 100’s of developers is difficult

A common solution is to use branches in SCM systems

- **Benefits:** Isolating concurrent work during times of instability

- **Cost:** Increase the time that changes percolate through the system (Code Velocity)
Status quo:
Many branches for little change

Code Velocity for this file is particularly bad…
Branch Analytics

Techniques:

- **Survey** devs to understand their problems with branching
- **Mine** dev. data for relationship of teams and branches
- **Simulate** benefits and cost of alternative branch structures

Actions/Tools:

- **Alert** users about possible conflicts
- **Recommend** branch structure, e.g. del., add, merge etc.
- **Perform** semi-automatic branch refactoring
Survey: Branching Problems

- **Big Bang Merge**: merge all branches simultaneously
- **Development Freeze**: stop work while merging
- **Integration Wall**: using branches between people, instead of dividing work
- **Branch Mania**: creating too many branches

Anti-patterns from *Streamed Lines: Branching Patterns for Parallel Software Development* and *Branching and Merging Primer*. 
Mine “File Similarity” / ”Developer Similarity”

Dark areas mean many branch pairs in that area.

Same devs working on different things is OK

Same files should mean same people

Most pairs of branches are not similar

Same files, but different team means possible problems
Simulate Cost-Benefit of Alternative Branch Structures

Idea: Replay Windows history
  • With each feature-branch removed

Measure impact on:
  • Velocity ("cost")
  • Avoided conflicts ("benefit")
Velocity vs. Conflict avoidance

- Bad branch
- Good branch
Summary: Branch Analytics

Software Analytics makes software development data actionable

- Branch analytics key to improve code velocity
- Better design of development structure
- Efficient scheduling
- Reliable systems with low conflicts
Equivalence Checking (EC)

THE BIG SUCCESS STORY OF FMS IN HARDWARE

Formally prove that two circuit designs, like register transfer level and netlist, exhibit exactly the same behavior.
EC for Software

Formally prove that two programs (with procedures) have the same input/output behavior

- Input: State of parameters, globals and heap
- Output: State of return, globals and heap
Example

void swap1(ref int x, ref int y){
    int z = x;
    x = y;
    y = z;
}

void swap2(ref int x, ref int y){
    x = x + y;
    y = x - y;
    x = x - y;
}

z0 == x0    &&
x1 == y0    &&
y1 == z0    &&
swap1.x == x1  &&  swap1.y == y1
&&
x1' == x0 + y0     &&
y1' == x1' − y0     &&
x2' == x1' − y1'    &&
swap2.x == x2'    &&  swap2.y == y1'
&&
~ (swap1.x == swap2.x &&
    swap1.y == swap2.y)
Example

void swap1(ref int x, ref int y){
    int z = x;
    x = y;
    y = z;
}
void swap2(ref int x, ref int y){
    x = x + y;
    y = x - y;
    x = x - y;
}

z0 == x0 &&
x1 == y0 &&
y1 == z0 &&
swap1.x == x1 && swap1.y == y1 &&
x1' == x0 + y0 &&
y1' == x1' - y0 &&
x2' == x1' - y1' &&
swap2.x == x2' && swap2.y == y1' &&
~ (swap1.x == swap2.x &&
    swap1.y == swap2.y)

Theorem prover

UNSAT (Equivalent)

SAT (Counterexample)
Interesting constructs in programs

- Branches
- Loops
- Heap and the stack
- Procedure calls
Procedure calls and uninterpreted functions

Two programs

void Foo1(ref int x, int y){
    int z = x + y;
    x = Bar(z);
}

void Foo2(ref int x, int y){
    int z = y + x;
    x = Bar(z);
}

Uninterpreted function

a == b \implies F\_Bar(a) == F\_Bar(b)

Formula/Constraint

z0 == x0 + y0 &&
    x1 == F\_Bar(z0) &&
    Foo1.x == x1

& &

z0' == x0 + x0 &&
    x1' == F\_Bar(z0') &&
    Foo2.x == x1'

& &

\neg (Foo1.x == Foo2.x)
SymDiff

A Semantic Diff tool

• Like Windiff

Language independent

• Builds on Boogie verifier and Z3 theorem prover

Adapt for various source languages

• C, C++, .NET, x86, ARM, ….
SymDiff for Applications and Compiler

Version/Application-compatibility:
Do the two versions behave the same?
Can be used to automatically resolve refactoring/bugfix conflicts

Translation Validation:
Do source and target program agree?

Version/Application-compatibility:
Do the two versions behave the same?
Can be used to automatically resolve refactoring/bugfix conflicts

Version/Application-compatibility:
Do the two versions behave the same?
Can be used to automatically resolve refactoring/bugfix conflicts
SymDiff tool

Works at Boogie intermediate language, i.e. generates verification conditions, which are discharged by the theorem prover Z3
SymDiff for C

Trace viewer

c:\tv\projects\symb\symdiff\symdiff\test\c_examples\ex5\v1\a.c:
1:    int g;
2:    int cmp (int r, int s)
3:    {
4:        int x;
5:    }
6:    int x;
7:    [x = 2, r = 1]
8:    [x = r - 2;
9:        [x = 2, r = 1]
10:    if (x & 0x0001)
11:        [x = 2]
12:        {
13:            x = r + s;
14:        } else
15:        {
16:            x = r - s;
17:        }
18:    g ++;
19:    [g = 4294967295]
20:    return x;
21:    [x = 0]


c:\tv\projects\symb\symdiff\symdiff\test\c_examples\ex5\v2\a.c:
1:    int g;
2:    int cmp (int r, int s)
3:    {
4:        int x;
5:    }
6:    int x;
7:    [x = r | 0x0001;
8:        [x = 1]
9:    if (x & 0x0001)
10:        [x = 1]
11:        {
12:            [x = 2, r = 1, s = 1]
13:        } else
14:        {
15:            [x = r - s;
16:        }
17:    [g = 4294967295]
18:    return x;
19:    [x = 2]
20:    }
Summary: SymDiff

Logic-based tools translate programs & constraints into formulas

SymDiff checks
- merge of refactorings
- application compatibility
- compiler translations
- “refinement” - the same except for undefined behavior

Try tools out yourself: http://rise4fun.com
• Over 1.8 billion people are connected to the web
• The browser is the most widely used app
• People love to play
Pex for Fun

Excite people to play coding duels (puzzles), i.e.

- Given a hidden program
- Puzzlers writes a user program
- Puzzler wins if hidden “=“ user, otherwise he gets counter examples

Enabled via Pex which uses dynamic symbolic execution, i.e. executes “all” paths of both programs
The code is a puzzle. Do you understand what the code does? Click Ask Pex! to find out.

```csharp
using System;

public class Program {
    // What values of v can cause an exception? Ask Pex to find out!
    public static void Puzzle(int[] v) {
        if (v != null &&
            v.Length > 0 &&
            v[3] == 12345) {
            throw new Exception("hidden bug!");
        }
    }
}
```

<table>
<thead>
<tr>
<th>v</th>
<th>Output/Exception</th>
<th>Error Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>null</td>
<td>Output/Exception</td>
<td>Error Message</td>
</tr>
<tr>
<td>{0}</td>
<td>Output/Exception</td>
<td>Error Message</td>
</tr>
<tr>
<td>{0}</td>
<td>IndexOutOfBoundsException</td>
<td>Index was outside the bounds of the array.</td>
</tr>
<tr>
<td>{0, 0, 0, 0}</td>
<td>IndexOutOfBoundsException</td>
<td>Index was outside the bounds of the array.</td>
</tr>
<tr>
<td>{0, 0, 0, 12345, 0}</td>
<td>Exception</td>
<td>hidden bug!</td>
</tr>
</tbody>
</table>
The code is a puzzle. Do you understand what the code does? Click Ask Pex! to find out.

```csharp
using System;

public class Program {
    public static void Puzzle() {
        Console.W
    }
}
```

- **Write (18)**
  - `Write (string value)`
  - Writes the specified string value, followed by the current line terminator, to the standard output stream.
  - **value**: The value to write.
  - **Throws**: `System.IO.IOException`
This puzzle is an interactive Coding Duel. Can you write code that matches a secret implementation? Other people have already won this Duel 1184 times! Help

using System;

public class Program {
    // Can you fill the puzzle method to match the secret arithmetic operation?
    public static int Puzzle(int x) {
        return 0;
    }
}

Ask Pex!

Done. 2 interesting inputs found. How does Pex work?

Pex found 1 difference between your puzzle method and the secret implementation. Improve your code, so that it matches the other implementation, and 'Ask Pex!' again. You are not signed in. Sign In to track your achievements. Help

<table>
<thead>
<tr>
<th>x</th>
<th>your result</th>
<th>secret implementation result</th>
<th>Output/Exception</th>
<th>Error Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>-1</td>
<td>Mismatch</td>
<td>Your puzzle method produced the wrong result.</td>
</tr>
</tbody>
</table>
Social Experience

205,324 clicked 'Ask Pex!'

User70933 asked Pex about a puzzle
11 seconds ago
User70933 tried to win C# - «ChallengeArithmetic1»
14 seconds ago
User70933 asked Pex about a puzzle
16 seconds ago
User70932 asked Pex about a puzzle
20 seconds ago
User70929 tried to win C# - «ChallengeDigits2»
32 seconds ago
User70929 tried to win C# - «ChallengeDigits2»
59 seconds ago
User70931 tried to win C# - «ChallengeArithmetic1»
2 minutes ago
User70918 made 17th attempt to win C# - «ChallengeWordReverse»
Teaching

The Social Classroom

Course Description: Learn how PexForFun turns teaching computer science into a social gaming experience.
Teacher: the Pex Team

Associated Pages:
- The Social Classroom Whitepaper
- The Social Classroom Sample Page

Registered Students:
- mbarnett
- Murray
- Kai
- (no nickname)
- Chris C Sharp
- (no nickname)
- TaoXie
- (no nickname)
- meisl
- (no nickname)
- Nima
- Rocky
- Micgi
- Ishtiaque

The following link allows any signed in user to register for and access this course.
http://pexforfun.com/thesocialclassroom

Status of The Social Classroom

Your Progress: Nikolai Tillmann, you already won 1 Coding Duel; 1 more to go!

Factorial 1 attempt
ArraySort won after 15 attempts

All Students’ Progress:

<table>
<thead>
<tr>
<th>Coding Duels</th>
<th>#0</th>
<th>#1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chris C Sharp</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>meisl</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Nima</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Micgi</td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

© 2010 Microsoft - Pex v0.94 - Terms of Use - Privacy
Pex for Fun: Conclusion

For reach, taking programming into the browser & cloud and make it fun

Pex4Fun opens opportunities for
• learning using coding duels
• socializing using live feed, sharing duels
• teaching with automatic grading
• research: recommend fixes based on 1/2 million attempts

Try it out yourself: http://pex4fun.com
In 2010, Smartphones outsold PCs
Today’s Smartphones more powerful than PC from 2000
But cannot be programmed …

Shouldn’t we change that?
Touchstudio

- Social experience of creating little apps
- On the phone for the phone and in the cloud

Examples:
- print “Hello world” should go to facebook
- set ring-tone based on GPS location
- filter twitter messages
- build your own media search
Users

teenagers

Excel macro-writers

you and everyone else
What’s needed?

Programming on the phone
• authoring, debugging, running

Easy access to sensors/services/apps
• discoverable, minimal amount of code

Social aspects
• share programs and their data with your friends

Cloud integration
• sharing state, split computation, privacy, security
Program Model

Program is a set of event-triggered, sequentially executed actions.

Async calls lead to automatic tomb-stoning and continuation.
Programming Environment

Semi-structured editing + calculator using touch

Programs on the phone; possibly shared as pictures with friends
Just for fun....

• Start location feature
• Create a bing map
• Get the current location
• Add a (‘here’) pushpin
• Geocode ‘portland, usa’
• Add a (‘to’) pushpin
• Calc. a route between the 2 points
• Display it on the map (green line)
• Take a screenshot,
• Save it to the library
Touchstudio Conclusion

Take programming on the Phone + Cloud

With TouchStudio research opportunities abound
• Dev.Environment: on the phone authoring/debugging/running
• Programming model: easy access to sensors/services/apps
• Cloud integration: sharing of programs & data, security, privacy
• Energy efficiency: on the phone/cloud, tier splitting

Try it out yourself: Windows Phone MarketStore
http://research.microsoft.com/Touchstudio
Software Analytics enables data-driven decision, i.e. which process, practice, tool to use and deploy under which context.

Logic based tools help develop better software artifacts, i.e. help model, analyze, optimize, and synthesize software artifacts.

Future platforms excite and pose new challenges, e.g. web, mobile devices (phone, tablet), datacenter, games.
Q & A

schulte@microsoft.com

http://www.rise4fun.com
http://www.pex4fun.com
http://research.microsoft.com/touchstudio
http://research.microsoft.com/rise/
http://research.microsoft.com/~schulte/