Software Engineering for Spreadsheets: Challenges and Opportunities

Martin Erwig
Oregon State University

Supported by NSF (ITR-0325273, CCF-0741584)
Motivation

- Spreadsheets are widely used
- Spreadsheets are prone to errors
- Spreadsheet errors have huge impact

Needed:

Efforts to make spreadsheets more dependable
Spectrum of Viewpoints

Application  Consulting, Development  Psychology  HCI  Programming Languages
Let’s Use Tools!

Processes is important ...

... but humans are fallible

Tools are indispensable in all areas of our daily lives
Contents

The Challenge

The Essence of Spreadsheet Computing

Opportunities for Tools

Example 1: Type Checking

Example 2: Debugging
Challenge

Editing Spreadsheets ≈ Software Engineering

Spreadsheet Users: End Users

Objective: get the job done

How to get end users to employ SE techniques?
Everybody is an End User
Contents

The Challenge

The Essence of Spreadsheet Computing

Opportunities for Tools

Example 1: Type Checking

Example 2: Debugging
The Essence of Spreadsheet Computing

\[ e ::= v \mid o(e, \cdots, e) \mid a \]

\[ s = a \rightarrow e \]
The Essence of Spreadsheet Computing

\[ e ::= \lambda x.e \mid e \, e \mid x \]

No Local Scope

No Higher-Order Operations ⇒ No Recursion
Contents

The Challenge

The Essence of Spreadsheet Computing

Opportunities for Tools

Example 1: Type Checking

Example 2: Debugging
Opportunities

- Limited Computation Model
- Spatial Embedding of Formulas

- No loops/recursion
- Simplified Reasoning for Tools
- Availability of Rich Contextual Information

[IEEE Software, Sept. 2009]
A Sweet Spot

- **Java, C, ...**
- **DSLs: SQL, ...**
- **Spreadsheets**

**Computation Model**
- **Complex**
- **Simple**

**Context Info**
- **Sparse**
- **Rich**
Contents

The Challenge
The Essence of Spreadsheet Computing
Opportunities for Tools
Example 1: Type Checking
Example 2: Debugging
End-User Type System?

**Problem**
Abstract concept of types is very difficult to convey to end users

**Idea**
Use vocabulary from the spreadsheet as type information

[PADL’02, HCC’02]
Type Checking

<table>
<thead>
<tr>
<th>Month</th>
<th>Fruit</th>
<th>Orange</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>May</td>
<td>Apple</td>
<td>17</td>
<td>9</td>
</tr>
<tr>
<td>June</td>
<td>Apple</td>
<td>8</td>
<td>13</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>25</td>
<td>22</td>
</tr>
</tbody>
</table>
Labeling Rules

\[ e : L \text{ and } L : K \Rightarrow e : K \rightarrow L \]

Fruit \( \rightarrow \) Apple

\[ e : L \text{ and } e : K \Rightarrow e : L \& K \]

Month \( \rightarrow \) May \& Fruit \( \rightarrow \) Apple

\[ e : L \text{ and } e' : K \Rightarrow e + e' : L | K \]

May Apple | May Orange
May (Apple | Orange)  Factoring
May Fruit  Generalization
A Summation Error

B5 = \text{SUM}(B2:B4)

- B2 : Fruit
- B3 : Fruit \rightarrow Apple
- B4 : Fruit \rightarrow Apple

B2's label cannot be factored ⇒ prevents generalization step
Reflections

• We can infer label relationships automatically [VL/HCC’04, JVLC’07]

• Users do use labels to reason about formulas in spreadsheets to debug effectively [VL/HCC’07]

• Combining syntactic label checking with semantic dimension analysis finds even more faults [VL/HCC’08a, JVLC’09, VL/HCC’09]

• Is it actually practical?
Part of Real-World Tool

### East Gorge District

<table>
<thead>
<tr>
<th>Year</th>
<th>Year 1</th>
<th>Year 2</th>
<th>Year 3</th>
<th>Year 4</th>
<th>Year 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cash inflow (Gross)</td>
<td>$ -</td>
<td>$ -</td>
<td>$ 60,000</td>
<td>$ 400,000</td>
<td>$ 620,000</td>
</tr>
<tr>
<td>COGS</td>
<td>$ -</td>
<td>$ -</td>
<td>$ (35,000)</td>
<td>$ (220,000)</td>
<td>$ (325,000)</td>
</tr>
<tr>
<td>Capital Expense</td>
<td>$ (500,000)</td>
<td>$ -</td>
<td>$ -</td>
<td>$ -</td>
<td>$ -</td>
</tr>
<tr>
<td>Adult Labor Cost</td>
<td>$ (20,000)</td>
<td>$ (20,000)</td>
<td>$ (20,000)</td>
<td>$ (25,000)</td>
<td>$ (25,000)</td>
</tr>
<tr>
<td>Depreciation</td>
<td>$ 116,600</td>
<td>$ 116,600</td>
<td>$ 116,600</td>
<td>$ 116,600</td>
<td>$ 17,000</td>
</tr>
<tr>
<td>Salvage</td>
<td>$ -</td>
<td>$ -</td>
<td>$ -</td>
<td>$ -</td>
<td>$ -</td>
</tr>
<tr>
<td>Net Cash Flow</td>
<td>$(503,400)</td>
<td>$(95,600)</td>
<td>$(120,800)</td>
<td>$(271,600)</td>
<td>$(433,600)</td>
</tr>
<tr>
<td>Converted (Dollars)</td>
<td>$ -</td>
<td>$ -</td>
<td>$ -</td>
<td>$ -</td>
<td>$ -</td>
</tr>
</tbody>
</table>

### North Park District

<table>
<thead>
<tr>
<th>Year</th>
<th>Year 1</th>
<th>Year 2</th>
<th>Year 3</th>
<th>Year 4</th>
<th>Year 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cash inflow (Gross)</td>
<td>€ -</td>
<td>€ -</td>
<td>€ 50,000</td>
<td>€ 350,000</td>
<td>€ 700,000</td>
</tr>
<tr>
<td>COGS</td>
<td>€ -</td>
<td>€ -</td>
<td>€ (30,000)</td>
<td>€ (190,000)</td>
<td>€ (350,000)</td>
</tr>
<tr>
<td>Capital Expense</td>
<td>€ (450,000)</td>
<td>€ -</td>
<td>€ -</td>
<td>€ -</td>
<td>€ -</td>
</tr>
<tr>
<td>Converted (Euros)</td>
<td>€ (229,000)</td>
<td>€ -</td>
<td>€ -</td>
<td>€ -</td>
<td>€ -</td>
</tr>
<tr>
<td>Adult Labor Cost</td>
<td>€ (20,000)</td>
<td>€ -</td>
<td>€ -</td>
<td>€ -</td>
<td>€ -</td>
</tr>
<tr>
<td>Depreciation</td>
<td>€ 116,600</td>
<td>€ -</td>
<td>€ -</td>
<td>€ -</td>
<td>€ -</td>
</tr>
<tr>
<td>Salvage</td>
<td>€ -</td>
<td>€ -</td>
<td>€ -</td>
<td>€ -</td>
<td>€ -</td>
</tr>
<tr>
<td>Net Cash Flow</td>
<td>€ (503,400)</td>
<td>€ (95,600)</td>
<td>€ (120,800)</td>
<td>€ (271,600)</td>
<td>€ (433,600)</td>
</tr>
<tr>
<td>Converted (Dollars)</td>
<td>$ (691,245)</td>
<td>$ (901,245)</td>
<td>$ (931,245)</td>
<td>$ (951,245)</td>
<td>$ (971,245)</td>
</tr>
</tbody>
</table>

### Project Overview

- Total Cash Flow: $(2,730,125)
- NPV at 10%: $(24,371)
Contents

The Challenge

The Essence of Spreadsheet Computing

Opportunities for Tools

Example 1: Type Checking

Example 2: Debugging
Goal-Directed Debugging

Key Ideas

• Reason directly from failure to fault
• Propagate change expectations to derive change suggestions
• Rank suggestions employing heuristics

[VL/HCC’05]

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6</td>
<td>A1+2</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

Computed 8

Expected 7

Change Suggestions

A1+2 → A1+1
A1+2 → A2+2
A1+2 → 7
Example

<table>
<thead>
<tr>
<th>Month</th>
<th>Fruit</th>
<th>Apple</th>
<th>Orange</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>May</td>
<td>Apple</td>
<td>17</td>
<td>9</td>
<td>30</td>
</tr>
<tr>
<td>June</td>
<td></td>
<td>8</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>25</td>
<td>22</td>
<td></td>
</tr>
</tbody>
</table>

Value Expectation

Value in D4 should be equal to 26
Example

$$D4 = B3 + C4$$

<table>
<thead>
<tr>
<th>Month</th>
<th>Fruit</th>
<th>Apple</th>
<th>Orange</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>May</td>
<td>Fruit</td>
<td>17</td>
<td>9</td>
<td>30</td>
</tr>
<tr>
<td>June</td>
<td>Fruit</td>
<td>8</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>25</td>
<td>22</td>
<td></td>
</tr>
</tbody>
</table>

In the formula: =B3+C4
- change B3 to C4
- change C4 to C3
Evaluation of GoalDebug

- Corrects 97% of faults
- Ranking of correct change suggestion

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>First</td>
<td>59%</td>
</tr>
<tr>
<td>Top 2</td>
<td>71%</td>
</tr>
<tr>
<td>Top 5</td>
<td>80%</td>
</tr>
</tbody>
</table>

[ICSE’07, VL/HCC’08b]

1. Introduce Mutation

2. Run GoalDebug

3. Find rank of correct change suggestions

\[ G \rightarrow \cdots \rightarrow G \rightarrow F \]
Conclusions

Simplified Computation Model & Spatial Embedding Facilitate Effective, Easy-to-Use Software Engineering Tools

More Examples:

• Generation of Correct Spreadsheet:
  Gencel [VL/HCC’05, ICSE’05, ICSE’06, JFP’06]
  ClassSheets [ASE’05, JOT’07]

• Testing [VL/HCC’06, VL/HCC’08, TSE’09]

• Pattern Inference [PPDP’06, ICSE’06]