Spreadsheet Engineering

Jácome Cunha
jacome@di.uminho.pt
www.di.uminho.pt/~jacome

Researcher @
HASLab / INESC TEC & Universidade do Minho, Portugal
Oregon State University (office 2077)

OSU - EECS Colloquium - 02/24/14
I. Motivation
II. Spreadsheets Meet Models
III. Models for Spreadsheets – ClassSheets
IV. Inferring ClassSheets
V. Embedding ClassSheets
VI. Evolution!
VII. Model-Driven Spreadsheets
VIII. Summary
I. Motivation
Why do Spreadsheets matter?

Financial intelligence firm CODA reports that 95% of all U.S. firms use spreadsheets for financial reporting.

_Sarbanes-Oxley: What About all the Spreadsheets?,_ Raymond R. Panko and Nicholas Ordway, 2008
Why do Spreadsheets matter?

They are the programming language of choice by non-professional programmers, a.k.a. end users.

In the U.S. alone, the number of end-user programmers is conservatively estimated at 11 million, compared to only 2.75 million other, professional programmers.

Estimating the numbers of end users and end-user programmers, Christopher Scaffidi, Mary Shaw, and Brad Myers, VL/HCC 2005
Omnipresent
Easy-to-use
Multi-purpose
Flexible
In 2004, RevenueRecognition.com (now Softtrax) had the International Data Corporation interview 118 business leaders.

IDC found that 85% were using spreadsheets in financial reporting and forecasting.

*Source: Sarbanes-Oxley: What About all the Spreadsheets?, Raymond R. Panko and Nicholas Ordway, 2008*
In fact, spreadsheets lack:

- Abstraction
- Encapsulation
- Type system
- Testing
- IDE
- ...

...
Around 200 people who thought their only experience of the London 2012 Olympic Games would be minor heats of synchronised swimming have received an unexpected upgrade to the men’s 100m final following an embarrassing ticketing mistake.

Locog said the error occurred in the summer, between the first and second round of ticket sales, when a member of staff made a single keystroke mistake and entered ‘20,000’ into a spreadsheet rather than the correct figure of 10,000 remaining tickets.
Harvard University economists Carmen Reinhart and Kenneth Rogoff have acknowledged making a spreadsheet calculation mistake in a 2010 research paper, “Growth in a Time of Debt”, which has been widely cited to justify budget-cutting.

Business Week, 18 April 2013

In a 2010 paper* Carmen Reinhart, now a professor at Harvard Kennedy School, and Kenneth Rogoff, an economist at Harvard University...argued that GDP growth slows to a snail’s pace once government-debt levels exceed 90% of GDP. The 90% figure quickly became ammunition in political arguments over austerity...This week a new piece of research poured fuel on the fire by calling the 90% finding into question..

The Economist, 17 April 2013

Economy losses of $10 billion/year!
II. Spreadsheets Meet Models
[PEPM'09, VL/HCC'10]
Why Models?
Spreadsheet Example

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Pilot-Id</td>
<td>Pilot-Name</td>
<td>Phone</td>
<td>Depart</td>
<td>Destination</td>
<td>Date</td>
<td>Hours</td>
<td>N-Number</td>
<td>Model</td>
</tr>
<tr>
<td>2</td>
<td>pl1</td>
<td>John</td>
<td>321654987 OPO</td>
<td>NAT</td>
<td>12/12/2010 – 14:00</td>
<td>07:00 N2342</td>
<td>B 747</td>
<td>Magalhães</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>pl2</td>
<td>Mike</td>
<td>147258369 OPO</td>
<td>NAT</td>
<td>01/01/2011 – 16:00</td>
<td>07:00 N2342</td>
<td>B 747</td>
<td>Magalhães</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>pl1</td>
<td>John</td>
<td>321654987 LIS</td>
<td>AMS</td>
<td>16/12/2010 – 10:00</td>
<td>02:45 N341</td>
<td>B 777</td>
<td>Cabral</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>pl3</td>
<td>John</td>
<td>469184201 OPO</td>
<td>CLJ</td>
<td>13/07/2013 – 10:00</td>
<td>05:45 N101</td>
<td>A 380</td>
<td>DSL</td>
<td></td>
</tr>
</tbody>
</table>
Functional Dependency?

\[
\begin{array}{|c|c|}
\hline
A & B \\
\hline
1 & a1 & b1 \\
2 & a1 & b1 \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
A & B \\
\hline
1 & a1 & b1 \\
2 & a1 & b2 \\
\hline
\end{array}
\]
Functional Dependencies

- We compute the business logic from the data, by inferring FDs

- They are the building blocks inferring models for (legacy) spreadsheets

- The better the FDs we infer, the better the model we compute!
Too Many??

["A"] -> ["B","C","D","E","F"]
["C"] -> ["A","B","D","E","F"]
["D"] -> ["A","B","C","E","F"]
["E"] -> ["A","B","C","D","F"]
["F"] -> ["A","B","C","D","E"]
["G"] -> ["H","I","J"]
["H"] -> ["G","I","J"]
["I"] -> ["G","H","J"]
["J"] -> ["G","H","I"]
["K"] -> ["L","M"]
["L"] -> ["K","M"]
["M"] -> ["K","L"]
["B","K"] -> ["A","C","D","E","F"]
["B","L"] -> ["A","C","D","E","F"]
["B","M"] -> ["A","C","D","E","F"]
Accidents happen

- We use a data mining algorithm which produces too many accidental FDs!

- We introduce some spreadsheet specific heuristics to filter out “accidental” FDs
Organize them

- **Label semantics**: often keys are labeled “code” or “id”
- **Label arrangement**: we prefer FDs respecting the order of columns
- **Antecedent size**: small keys are preferable
- **Ratio**: small ratio between keys and non-keys
- **Single value columns**: columns always with the same value appear in too many FDs
## Final set

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>pl1</td>
<td>John</td>
<td>321654987 OPO</td>
<td>NAT</td>
<td>12/12/2010 – 14:00</td>
<td>07:00 N2342</td>
<td>B</td>
<td>747</td>
<td>Magalhães</td>
</tr>
<tr>
<td>3</td>
<td>pl2</td>
<td>Mike</td>
<td>147258369 OPO</td>
<td>NAT</td>
<td>01/01/2011 – 16:00</td>
<td>07:00 N2342</td>
<td>B</td>
<td>747</td>
<td>Magalhães</td>
</tr>
<tr>
<td>4</td>
<td>pl1</td>
<td>John</td>
<td>321654987 LIS</td>
<td>AMS</td>
<td>16/12/2010 – 10:00</td>
<td>02:45 N341</td>
<td>B</td>
<td>777</td>
<td>Cabral</td>
</tr>
<tr>
<td>5</td>
<td>pl3</td>
<td>John</td>
<td>469184201 OPO</td>
<td>CLJ</td>
<td>13/07/2013 – 10:00</td>
<td>05:45 N101</td>
<td>A</td>
<td>380</td>
<td>DSL</td>
</tr>
</tbody>
</table>

Pilot-Id → Pilot-Name, Phone

N-Number → Model, Plane-Name

Pilot-Id, N-Number, Depart, Destination, Date, Hours → {}
Having computed the FDs, we can now use the FUN algorithm to produce a relational model for the spreadsheet:

*Pilots* (Pilot-Id, Pilot-Name, Phone)

*Planes* (N-Number, Model, Plane-Name)

*<Flights>* (#Pilot-Id,# N-Number, Depart, Destination, Date Hours)
III. Models for Spreadsheet – ClassSheets
Engels and Erwig ASE'05
ClassSheets are a high-level, object-oriented formalism to specify spreadsheets

(a) Pilots’ table.

(b) Pilots’ visual ClassSheet model.

(c) Pilots’ textual ClassSheet model.
ClassSheets - Models for Spreadsheets

(a) Planes’ table.

(b) Planes’ visual ClassSheet model.

(c) Planes’ textual ClassSheet model.
(a) Flights’ visual ClassSheet model.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flights</td>
<td>PlanesKey</td>
<td></td>
<td></td>
<td></td>
<td>Plane_key=Planes.n-number</td>
</tr>
<tr>
<td>PilotsKey</td>
<td>Depart =&quot;&quot;</td>
<td>Destination =&quot;&quot;</td>
<td>Date =d</td>
<td>Hours =0</td>
<td>Total Pilot Hours =SUM(hours)</td>
</tr>
<tr>
<td></td>
<td>pilot_key=Pilots.ID</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(b) Flights’ table.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flights</td>
<td>PlanesKey</td>
<td>Plane_key=Planes.n-number</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>N2342</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PilotsKey</td>
<td>Depart =&quot;&quot;</td>
<td>Destination =&quot;&quot;</td>
<td>Date =d</td>
<td>Hours =0</td>
<td>Total Pilot Hours =SUM(PilotsKey.total)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>pilot_key=Pilots.ID</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>pl1</td>
<td>OPO</td>
<td>NAT</td>
<td>12/12/2010 – 14:00</td>
<td>07:00</td>
<td></td>
<td></td>
<td></td>
<td>09:45</td>
</tr>
<tr>
<td></td>
<td>pl2</td>
<td>OPO</td>
<td>NAT</td>
<td>01/01/2011 – 16:00</td>
<td>07:00</td>
<td></td>
<td></td>
<td></td>
<td>07:00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>16:45</td>
<td></td>
</tr>
</tbody>
</table>
IV. Inferring ClassSheets
[VL/HCC'10]
ClassSheet Inference

Original SS

Detect FDs

A → B
C D → E

Infer CS

Improved SS

Generate SS App

A

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><strong>Income</strong></td>
</tr>
<tr>
<td>2</td>
<td><strong>Item</strong></td>
</tr>
<tr>
<td>3</td>
<td><strong>value = 0</strong></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td><strong>Total</strong></td>
</tr>
<tr>
<td>5</td>
<td>total = SUM(Item.value)</td>
</tr>
</tbody>
</table>
V. Embedding ClassSheets
[VL/HCC'11]
Why the Embedding?
Embedding a language into another language is a recurring strategy (e.g. for DSLs)
- Embedded language inherit all the power of the host language :-)
- Users are used to the host language and do not need to learn a (complete) new language :-)
- Implementation effort is much reduced :-)
- It may have some restrictions :-(

We embedded ClassSheets in traditional spreadsheet systems
Vertically Expandable Tables

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>Flight hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Pilots</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>ID</td>
<td>Name</td>
</tr>
<tr>
<td>3</td>
<td>pl1</td>
<td>John</td>
</tr>
<tr>
<td>4</td>
<td>pl2</td>
<td>Mike</td>
</tr>
<tr>
<td>5</td>
<td>pl3</td>
<td>Anne</td>
</tr>
</tbody>
</table>

```
Horizontally Expandable Tables

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Planes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>N-Number</td>
<td>N2342</td>
<td>N341</td>
<td>N1343</td>
</tr>
<tr>
<td>3</td>
<td>Model</td>
<td>B 747</td>
<td>B 777</td>
<td>A 380</td>
</tr>
<tr>
<td>4</td>
<td>Name</td>
<td>Magalhães</td>
<td>Cabral</td>
<td>Nunes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Planes</td>
<td>n-number=&quot;&quot;</td>
</tr>
<tr>
<td>2</td>
<td>N-Number</td>
<td>model=&quot;&quot;</td>
</tr>
<tr>
<td>3</td>
<td>Model</td>
<td>name=&quot;&quot;</td>
</tr>
<tr>
<td>4</td>
<td>Name</td>
<td></td>
</tr>
</tbody>
</table>
### Relationship Tables

**Flights**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flights</td>
<td>PlanesKey</td>
<td></td>
<td></td>
<td>PlanesKey</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>N2342</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**PilotsKey**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
</tr>
</thead>
<tbody>
<tr>
<td>PilotsKey</td>
<td>Depart</td>
<td>Destination</td>
<td>Date</td>
<td>Hours</td>
<td>Depart</td>
<td>Destination</td>
<td>Date</td>
<td>Hours</td>
<td>Total Pilot Hours</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>pl1</td>
<td>OPO</td>
<td>NAT</td>
<td>07:00</td>
<td>LIS</td>
<td>AMS</td>
<td>16/12/2010</td>
<td>02:45</td>
<td>09:45</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>pl1</td>
<td>OPO</td>
<td>NAT</td>
<td>01/01/2011</td>
<td>07:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Flights**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flights</td>
<td>PlanesKey</td>
<td>plane_key=Planes.n-number</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**PilotsKey**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>PilotsKey</td>
<td>Depart</td>
<td>Destination</td>
<td>Date</td>
<td>Hours</td>
<td>Total Pilot Hours</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>total=SUM(hours)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>total=SUM(hours)</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>total=SUM(hours)</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>total=SUM(PlanesKey.total)</td>
<td></td>
</tr>
</tbody>
</table>
The foreign key constraint guarantees that only existing values are referenced, or created if necessary.

The value constraint guarantees that values of a block are within a certain range.

The primary key constraint guarantees that no duplicate values are inserted in a block.
VI. Evolution!
[FASE'11, ICMT'12]
Suppose now you need to add new information to the spreadsheet.

For instance, the number of passengers of each flight.

It would require to do several error-prone tasks:

- Add columns, labels, update formulas, etc.

We can do it automatically!
Why do Spreadsheet Instances Need Evolution?

- Some evolution steps are easier to perform on the instance
- For instance, to add a column to one of the repetition blocks
- People felt the need to evolve the data
### Flights

<table>
<thead>
<tr>
<th>Flights</th>
<th>PlanesKey</th>
<th>Depart</th>
<th>Destination</th>
<th>Date</th>
<th>Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>plane_key=Planes.N-Number</td>
<td>depart=&quot;&quot;</td>
<td>destination=&quot;&quot;</td>
<td>date=01/01/2010 - 00:00</td>
<td>hours=1 &gt; 0</td>
</tr>
</tbody>
</table>

Total = SUM(hours)

### Pilots

<table>
<thead>
<tr>
<th>ID*</th>
<th>Name</th>
<th>Flight hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>John</td>
<td>3400</td>
</tr>
<tr>
<td>p2</td>
<td>Mike</td>
<td>330</td>
</tr>
<tr>
<td>p3</td>
<td>Anne</td>
<td>433</td>
</tr>
</tbody>
</table>

### Planes

<table>
<thead>
<tr>
<th>N-Number</th>
<th>Model</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>N2342</td>
<td>B 747</td>
<td>Magalhães</td>
</tr>
<tr>
<td>N341</td>
<td>B 777</td>
<td>Cabral</td>
</tr>
<tr>
<td>N1343</td>
<td>A 380</td>
<td>Nunes</td>
</tr>
</tbody>
</table>

=> data co-evolution

model evolution

=> model co-evolution

data evolution
Bidirectional Transformation System

ClassSheet \( \xrightarrow{Op_M} \) ClassSheet'

Spreadsheet \( \xleftarrow{\text{conforms to}} \) Spreadsheet'

\( \xrightarrow{Op_D} \)
(Data) Operations on Instances

\[
data \ \text{Op}_D : \text{Data} \to \text{Data} =
\]

- \text{addColumn}_D \quad \text{Where} \quad \text{Index} \quad \quad \text{-- add a column}
- \text{delColumn}_D \quad \text{Index} \quad \quad \text{-- delete a column}
- \text{addRow}_D \quad \text{Where} \quad \text{Index} \quad \quad \text{-- add a row}
- \text{delRow}_D \quad \text{Index} \quad \quad \text{-- delete a row}
- \text{AddColumn}_D \quad \text{Where} \quad \text{Index} \quad \quad \text{-- add a column to all instances}
- \text{DelColumn}_D \quad \text{Index} \quad \quad \text{-- delete a column from all instances}
- \text{AddRow}_D \quad \text{Where} \quad \text{Index} \quad \quad \text{-- add a row to all instances}
- \text{DelRow}_D \quad \text{Index} \quad \quad \text{-- delete a row from all instances}
- \text{replicate}_D \quad \text{ClassName} \quad \text{Direction} \quad \text{Int} \quad \text{Int} \quad \quad \text{-- replicate a class}
- \text{addInstance}_D \quad \text{ClassName} \quad \text{Direction} \quad \text{Model} \quad \quad \text{-- add a class instance}
- \text{setLabel}_D \quad (\text{Index}, \text{Index}) \quad \text{Label} \quad \quad \text{-- set a label}
- \text{setValue}_D \quad (\text{Index}, \text{Index}) \quad \text{Value} \quad \quad \text{-- set a cell value}
- \text{SetLabel}_D \quad (\text{Index}, \text{Index}) \quad \text{Label} \quad \quad \text{-- set a label in all instances}
- \text{SetValue}_D \quad (\text{Index}, \text{Index}) \quad \text{Value} \quad \quad \text{-- set a cell value in all instances}
(Model) Operations on ClassSheets

\[
\text{data } Op_M : \text{Model } \rightarrow \text{Model } = \\
\begin{align*}
\text{addColumn}_M & : \text{Where Index} & \quad & \text{-- add a new column} \\
\text{delColumn}_M & : \text{Index} & \quad & \text{-- delete a column} \\
\text{addRow}_M & : \text{Where Index} & \quad & \text{-- add a new row} \\
\text{delRow}_M & : \text{Index} & \quad & \text{-- delete a row} \\
\text{setLabel}_M & : (\text{Index, Index}) \text{ Label} & \quad & \text{-- set a label} \\
\text{setFormula}_M & : (\text{Index, Index}) \text{ Formula} & \quad & \text{-- set a formula} \\
\text{replicate}_M & : \text{ClassName Direction Int Int} & \quad & \text{-- replicate a class} \\
\text{addClass}_M & : \text{ClassName (Index, Index) (Index, Index)} & \quad & \text{-- add a static class} \\
\text{addClassExp}_M & : \text{ClassName Direction (Index, Index) (Index, Index)} & \quad & \text{-- add an expandable class}
\end{align*}
\]
Bidirectional Transformation Functions

\[
\begin{align*}
\text{to : } & \text{Model } \times \text{Op}_M \rightarrow \text{Op}^*_D \\
to (\text{addColumn}_M) & \text{ w i } = \text{AddColumn}_D \text{ w (columnIndex}_D \text{ i)} \\
to (\text{delColumn}_M) & \text{ w i } = \text{DelColumn}_D \text{ (columnIndex}_D \text{ i)} \\
to (\text{addRow}_M) & \text{ w i } = \text{AddRow}_D \text{ w (rowIndex}_D \text{ i)} \\
to (\text{delRow}_M) & \text{ w i } = \text{DelRow}_D \text{ (rowIndex}_D \text{ i)} \\
to (\text{setLabel}_M) & (i, j) l = \text{setLabel}_D \text{ (position}_D \text{ (i, j)) l} \\
to (\text{setFormula}_M) & (i, j) f = \text{setValue}_D \text{ (position}_D \text{ (i, j)) f}
\end{align*}
\]

\[
\begin{align*}
\text{from : } & \text{Op}_D \rightarrow \text{Op}^*_M \\
\text{from (addColumn}_D \text{ w i) =} & \text{replicate}_M \text{ className Horizontal classInstances instanceIndex}_M \\
& \text{; addColumn}_M \text{ w columnOffsetIndex}_M \\
\text{from (delColumn}_D \text{ i) =} & \text{replicate}_M \text{ className Horizontal classInstances instanceIndex}_M \\
& \text{; delColumn}_M \text{ columnOffsetIndex}_M \\
\text{from (addRow}_D \text{ w i) =} & \text{replicate}_M \text{ className Vertical classInstances rowIndex}_M \\
& \text{; addRow}_M \text{ w rowOffsetIndex}_M \\
\text{from (delRow}_D \text{ i) =} & \text{replicate}_M \text{ className Vertical classInstances rowIndex}_M \\
& \text{; delRow}_M \text{ rowOffsetIndex}_M \\
\text{from (setLabel}_D \text{ (i, j) l) =} & \text{replicate}_M \text{ className Horizontal classInstances columnIndex}_M \\
& \text{; replicate}_M \text{ className Vertical classInstances rowIndex}_M \\
& \text{; setLabel}_M \text{ positionOffset}_M \text{ l} \\
\text{from (setValue}_D \text{ (i, j) l =} & \emptyset \\
\text{from (addInstance}_D \text{ cn dir m) =} & \emptyset
\end{align*}
\]
Compositional Example: *Add a Column and a Class*

addRow\textsubscript{M} Before 3; addClassExp\textsubscript{M} "BlueClass" Horizontal (2,1) (3,4)

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$\Rightarrow$

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$\Rightarrow$
VII. MDSheet – Model-Driven Spreadsheets
[ICSE'12]
The foreign key constraint guarantees that only existing values are referenced, or created if necessary.

The value constraint guarantees that values of a block are within a certain range.

The primary key constraint guarantees that no duplicate values are inserted in a block.

MDSheet Tool
http://youtu.be/6LNdTdCpV2U
• Available at http://ssaapp.di.uminho.pt

• Built out of 7886 LOC:
  - 3181 in Haskell, for the inference and evolution
  - 980 in Basic, for the embedding
  - 2884 in C++, for gluing all components
  - 340 in Perl, for compilation and setup
  - 722, for makefiles
Spreadsheets are great!

Omnipresent
Easy
Multi-purpose
Flexible

And the consequences may be...

Relational Model

- Having computed the FDs, we can now use the FUN algorithm to produce a relational model for the spreadsheet:

I. ClassSheet Model Inference

Embedding ClassSheets in Spreadsheets

Bidirectional Transformation System

MDSheet Tool
http://www.eusprig.org/stories.htm
Acknowledgments

This work has been done in collaboration with many people:

Martin Erwig, João Paulo Fernandes, Jorge Mendes, Hugo Pacheco, Rui Pereira, João Saraiva, Joost Visser
Thanks!

Questions?
More?

- More at http://ssaapp.di.uminho.pt
- Querying model-driven spreadsheet
- Visually querying model-driven spreadsheets
- Detections of bad smells
- Edit assistance
- Empirical validations
- Variational spreadsheets (@ OSU)
- ...

...
Does It Work?
Empirical Study Settings

- 17 student from a MSc course
- 2 different spreadsheets
  - Microsoft budget
  - Local company responsible for water supply of Braga, Portugal - agere
Hypotheses:

(1) In order to perform a given set of tasks, users spend less time when using model-driven spreadsheets instead of plain ones.

(2) Spreadsheets developed in the model-driven environment hold less errors than plain ones.
Main Results

Number of tasks performed on the MS spreadsheet

![Bar chart showing completion rates for different participants using plain and model-driven methods.](chart.png)
Main Results

Error rate in the budget spreadsheet