Embedding and Evolution of Spreadsheet Models in Spreadsheet Systems

Jácome Cunha, Jorge Mendes, João Saraiva

Universidade do Minho

Portugal

João Paulo Fernandes

Universidade do Minho & Universidade do Porto

Portugal

HASLab – October 26, 2011

Agenda

- Introduction
- Embedding ClassSheets into Spreadsheet Systems
- Co-Evolution of Spreadsheet Models and Data
- Conclusions and Future Work

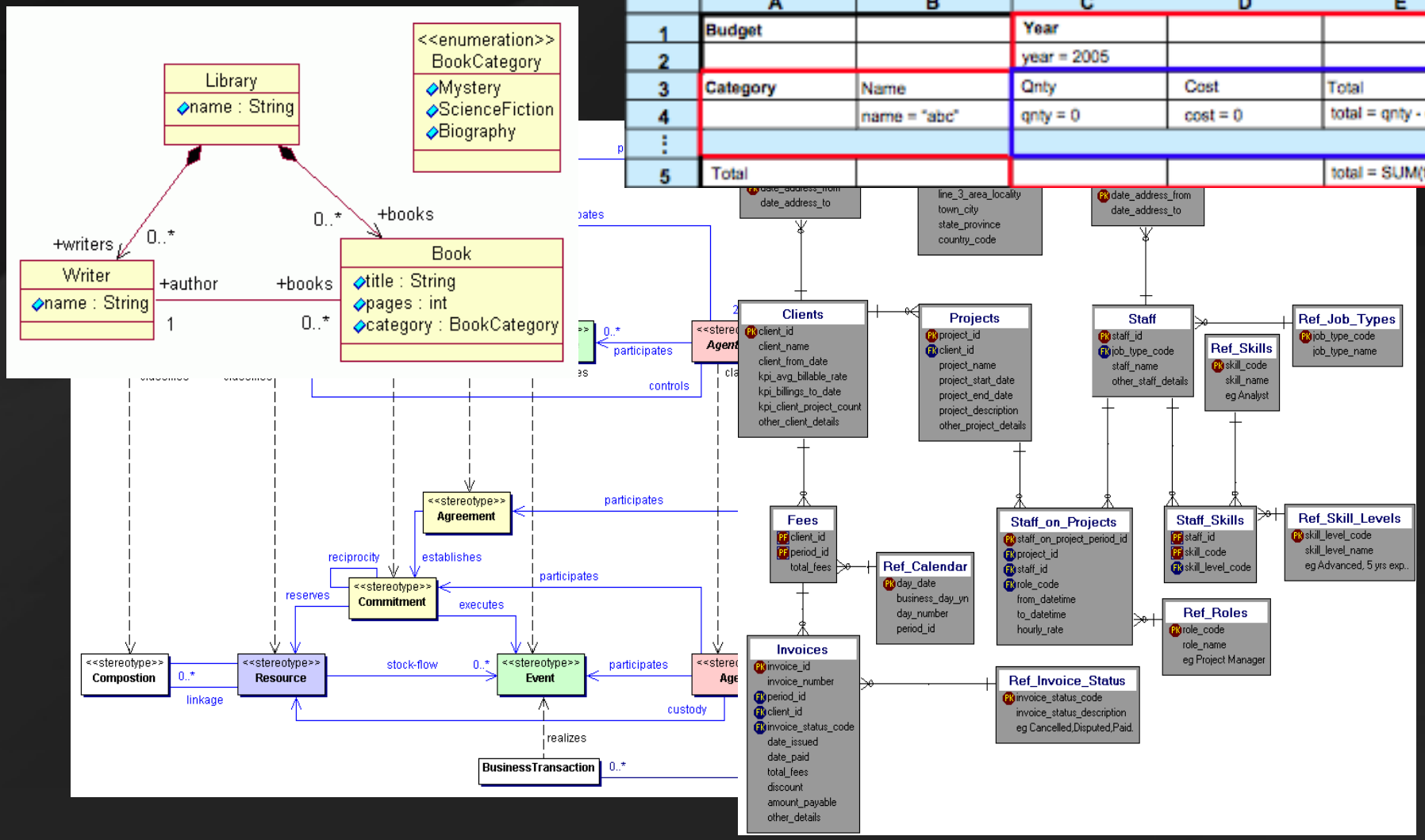
Introduction

Spreadsheets are widely used

Image taken from http://www.flickr.com/photos/cosmosfan/2414002070/

Spreadsheets contain many errors

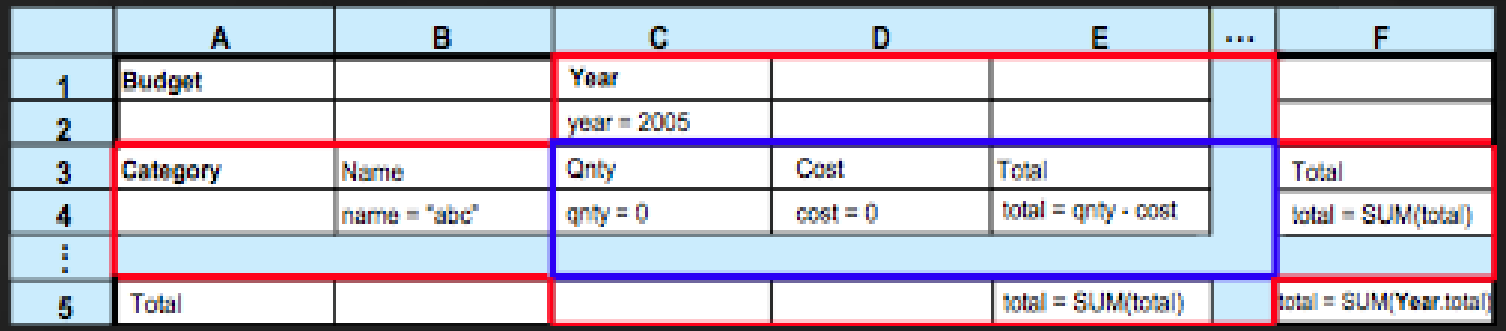
Image taken from http://www.computerrepairmaintenance.com/secrets-to-fixing-computer-problems/



	A	B	C	D	E	...	F
1	Budget		Year				
2			year = 2005				
3	Category	Name	Qnty	Cost	Total		Total
4		name = "abc"	qnty = 0	cost = 0	total = qnty - cost		total = SUM(tot
⋮							
5	Total				total = SUM(total)		total = SUM(Year

Model-based approach promises good results

Images taken from http://www.getopt.org/ecimf/contrib/onto/REA/REA-UML.gif and http://www.telecom-lille1.eu/people/Vanwormhoudt/siteEMFOCL/tutoriel/TutorialEMFOCL1.htm and http://weblogs.asp.net/scottgu/archive/2006/07/12/Tip_2F00_Trick_3A00_-Online-Database-Schema-Samples-Library.aspx



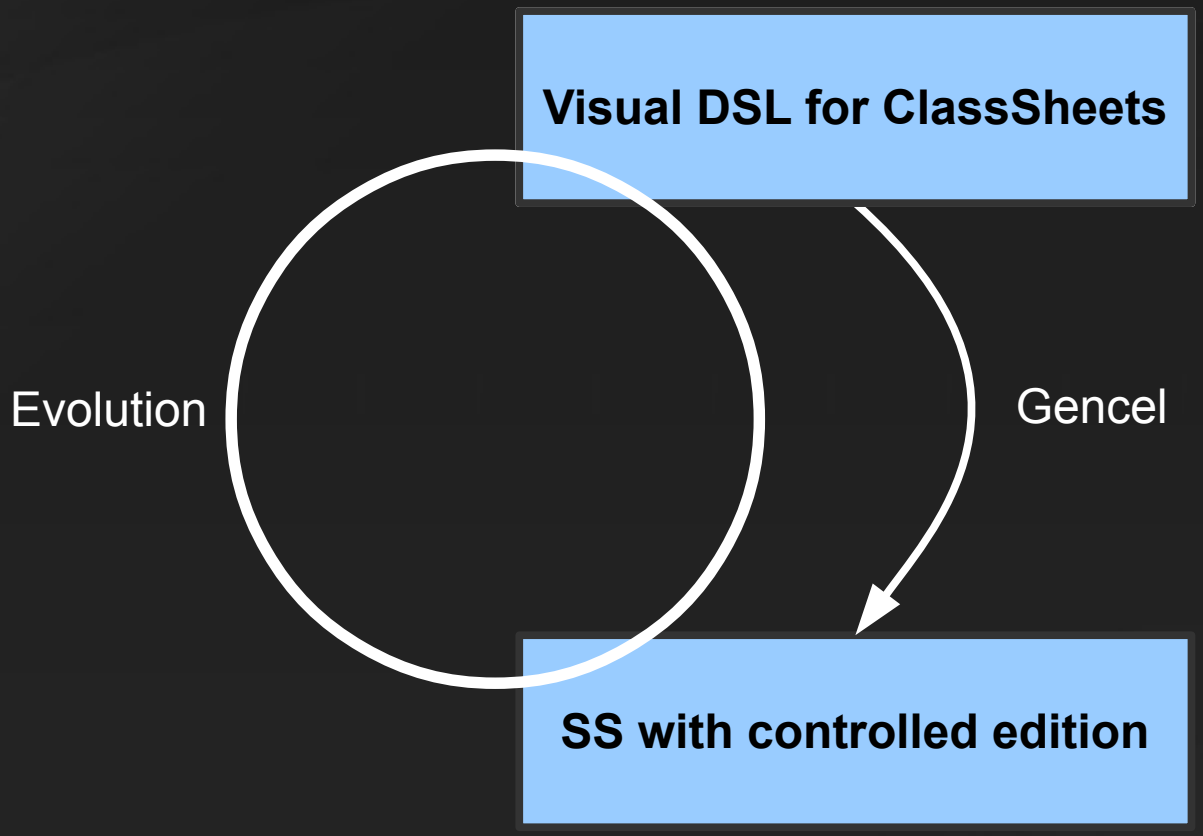
	A	B	C	D	E	…	F
1	Budget		Year				
2			year = 2005				
3	Category	Name	Qnty	Cost	Total		Total
4		name = "abc"	qnty = 0	cost = 0	total = qnty - cost		total = SUM(total)
⋮							
5	Total				total = SUM(total)		total = SUM(Year.total)

ClassSheets to the rescue!

Embedding ClassSheets into Spreadsheet Systems

Why Embedding?

- *Gencel* generates Excel spreadsheets
- Similar approach as compilers
- It makes it impossible to have synchronized evolution of both the model and the data
- We do not follow this compiler approach
- Instead, we use the embedding so we can reuse the Excel functionalities
- The management becomes easier: both the model and the data in the same environment


Visual DSL for ClassSheets
Evolution
Gencel
SS with controlled edition

Vertically Expandable Tables

	A	B	C
1	**Pilots**		
2	**ID**	**Name**	**Flight hours**
3	pl1	John	3400
4	pl2	Mike	330
5	pl3	Anne	433

	A	B	C
1	**Pilots**		
2	**ID**	**Name**	**Flight hours**
3	id=""	name=""	flight_hours=0
4	⋮	⋮	⋮

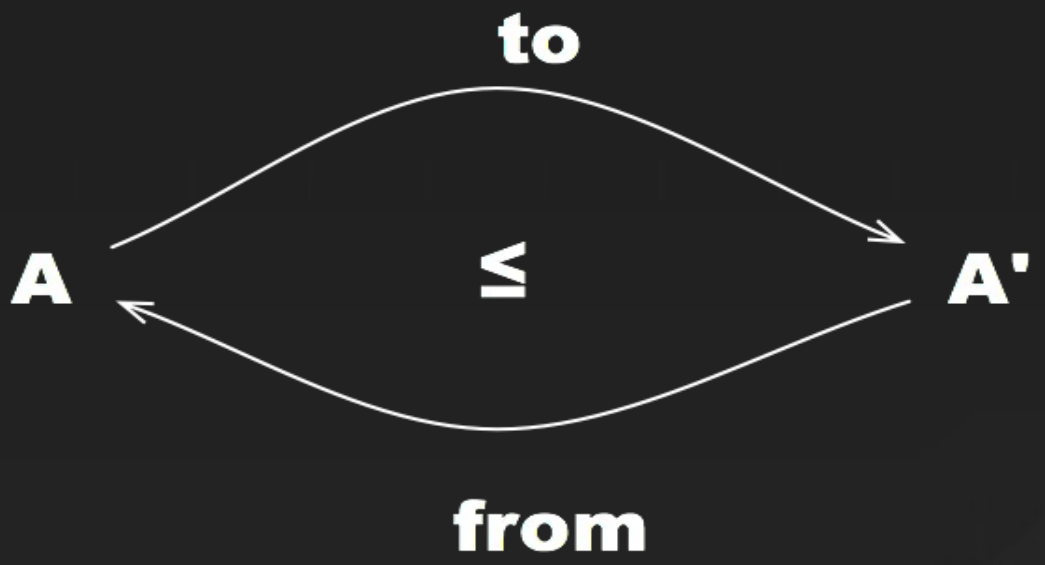
Relationship Tables

	A	B	C	D	E	F	G
1	**Flights**	**PlanesKey**				…	
2		plane_key=**Planes**.n-number				…	
3	**PilotsKey**	**Depart**	**Destination**	**Date**	**Hours**	…	**Total Pilot Hours**
4	pilot_key=**Pilots.ID**	depart=""	destination=""	date=d	hours=0	…	total=SUM(hours)
5	⋮	⋮	⋮	⋮	⋮	…	⋮
6					total=SUM(hours)	…	total=SUM(**PlanesKey**.total)

	A	B	C	D	E	F	G	H	I	J	K
1	**Flights**	**PlanesKey**				**PlanesKey**					
2		N2342				N341					
3	**PilotsKey**	**Depart**	**Destination**	**Date**	**Hours**	**Depart**	**Destination**	**Date**	**Hours**		**Total Pilot Hours**
4	pl1	OPO	NAT	12/12/2010 – 14:00	07:00	LIS	AMS	16/12/2010 – 10:00	02:45	…	09:45
5	pl1	OPO	NAT	01/01/2011 – 16:00	07:00						07:00
6	⋮										
7					14:00				02:45		16:45

Co-Evolution of Spreadsheet Models and Data

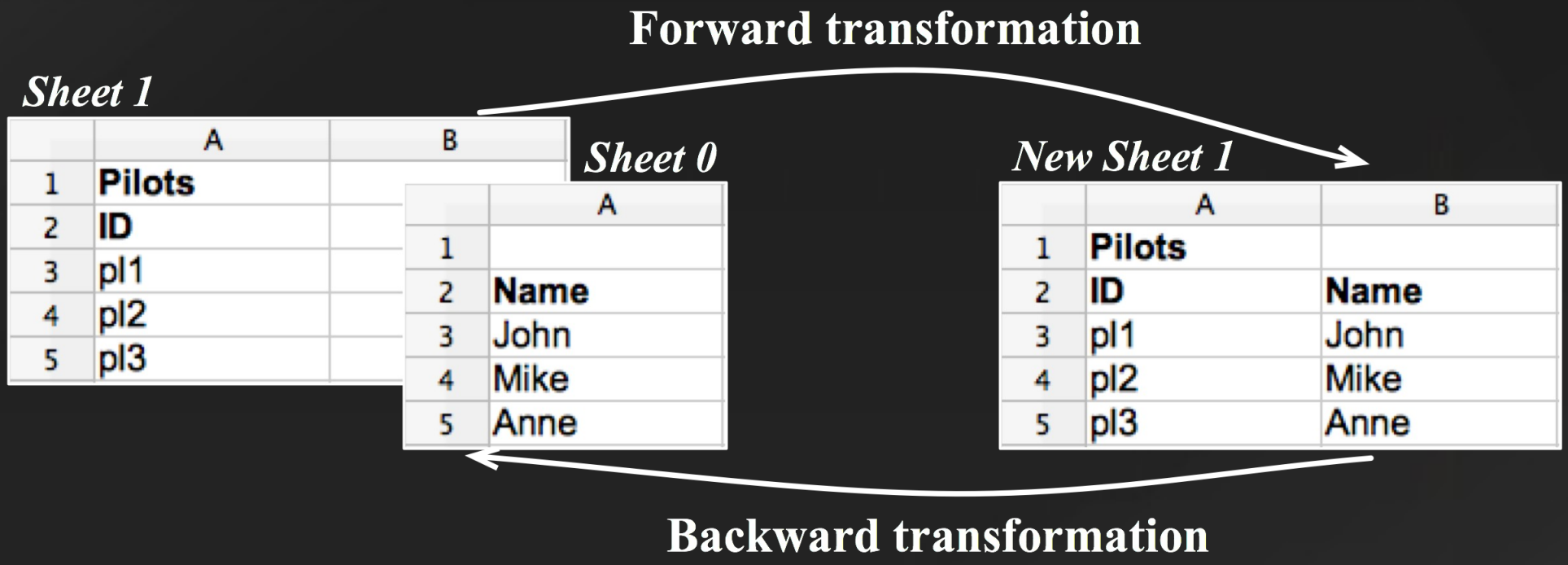
Data Refinements - 2LT



Co-Evolution Rules

- **Combinator rules**: *after*, *before*, *at*
- **Semantic rules**: *insert a column*, *make it expandable*
- **Layout rules**: *change orientation (transpose)*

Add/Remove Column Rule



Sheet 1

	A	B
1	**Pilots**	
2	**ID**	
3	pl1	
4	pl2	
5	pl3	

Sheet 0

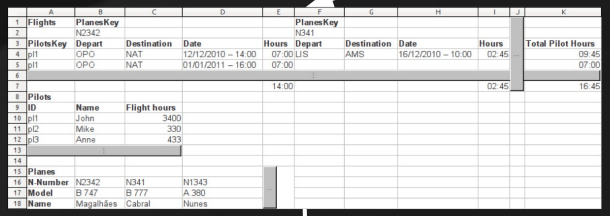
	A
1	
2	**Name**
3	John
4	Mike
5	Anne

New Sheet 1

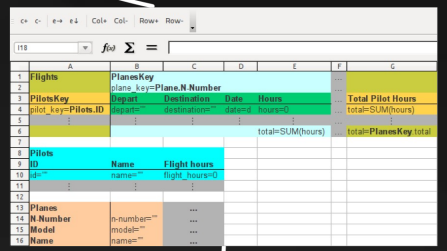
	A	B
1	**Pilots**	
2	**ID**	**Name**
3	pl1	John
4	pl2	Mike
5	pl3	Anne

From the model we can generate a template

Sheet 1



Sheet 0



Sync

Button pressed

OOBasic sends sheet 1 (data) to HaExcel the back-end

OOBasic sends sheet 0 (model) to the HaExcel back-end

Haskell spreadsheet representation

Haskell CS data type

Application of evolution rule chosen by the end user

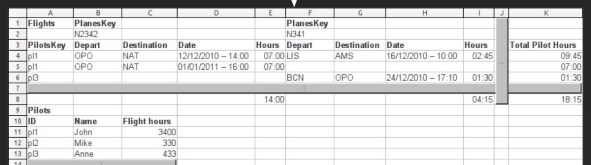
Forward and backward transformations

New Haskell CS data type

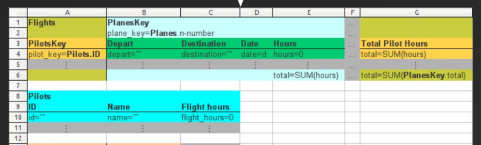
Application of the forward/backward tansformation

New Haskell spreadsheet representation

Sheet 1



Sheet 0



Sync

Conclusions

- We have shown how to embed a visual DSL into a traditional spreadsheet system
- This allows user to create models and instances in the same environment
- We used a formal framework to design and implement evolution steps
- The model and its instances are always synchronized

Future Work (in Progress)

- Extend the ClassSheet model with restrictions
  - mark=0:[0..20]
  - studentID=(a|pg|id)\d+
  - status={WORKER-STUDENT, REGULAR}
- Allow users to change data and infer the “best” evolved ClassSheet model