Smart Sheets

A new breed of spreadsheet computing

Enzo Alda – Principal and Founder – Lakebolt Research

Spreadsheets

- Decades old and still prevalent
- Great programming tool:
  - Visual
  - Functional
  - Reactive
- Most used functional programming tool
  - Why are they so successful?
- Prevalent spreadsheets
  - MS-Excel
  - Google Spreadsheets

A Brief History of Spreadsheets

- Pre-cambrian
- Cambrian explosion
  - Visicalc
  - Lotus 1-2-3
  - MS-Excel
- The great extinction
- Cloud Era
  - Microsoft Excel 360
  - Google Spreadsheet
  - ZOHO

Current State of the Art

- MS-Excel is the “reference implementation”
- Acceptance of basic design assumptions
- Everyone tries to be “just like Excel”
  - WARNING: repetitive (boring) slides ahead!
    - Quick survey of rants about spreadsheets from the Web

Issues:

from the Web

- **Source:** ANKHOR feel your data flow
- **Problem 1: Difficult inspection / lack of traceability**: In complex spreadsheets it is very hard to keep the overview concerning the formulas. Auditing the correctness of a spreadsheet can become a cumbersome task, especially in documents with several sheets making cross-references all over the place.
- **Problem 2: Inflexibility:** It is hard to rearrange the solution when it turns out that the trial-and-error development process went into the wrong direction and needs to be corrected.
- **Problem 3: Limited reuse**: Applying a solution to slightly different source data sets is difficult. A spreadsheet created for one purpose can in most cases not be reused for even a slightly different problem. It is not possible to create libraries with subsets of a solution and rearrange or recombine those easily for application to a new problem.
- **Problem 4: Limited scalability**: Processing very large amounts of data in a spreadsheet causes problems as the complete source data has to be present at once – there is no simple way for "streaming" large amounts of data through the spreadsheet solution.
- **Problem 5: Deployment**: It is not easily possible to turn a spreadsheet into a standalone solution that can be deployed as a distributable product or be published as a web site.

Issues:

from the Web

- Source: DENIZON
- Susceptibility to trivial manual errors
  - Due to the fundamental structure of spreadsheets, a slight change in the formula or value in any of their inhabited cells may already affect their overall output: e.g. accidental copy-paste, erroneous range selection, incorrect data input or unintentional deletion of a character, cell, range, column, or row
- Possibility of the user working on the wrong version
  - Since the most common reports are usually generated on a monthly basis, users tend to store them using file/directory names variations. A user can accidentally work on the wrong version.
- Prone to inconsistent company-wide reporting
- Often defenseless against unauthorized access
- Highly vulnerable to fraud
- Spreadsheet risk mitigation solutions may not suffice
  - No clear ownership of risk management responsibilities

- **To get rid of spreadsheet risks, you'll have to get rid of spreadsheets altogether**

Issues:

from the Web

- Source: DENIZON
- 1. Vulnerable to fraud
- 2. Susceptible to trivial human errors
- 3. Difficult to troubleshoot or test
- 4. Obstructive to regulatory compliance
- 5. Unfit for agile business practices
- 6. Not designed for collaborative work
- 7. Hard to consolidate
- 8. Incapable of supporting quick decision making
- 9. Unsuited for business continuity
- 10. Scales poorly

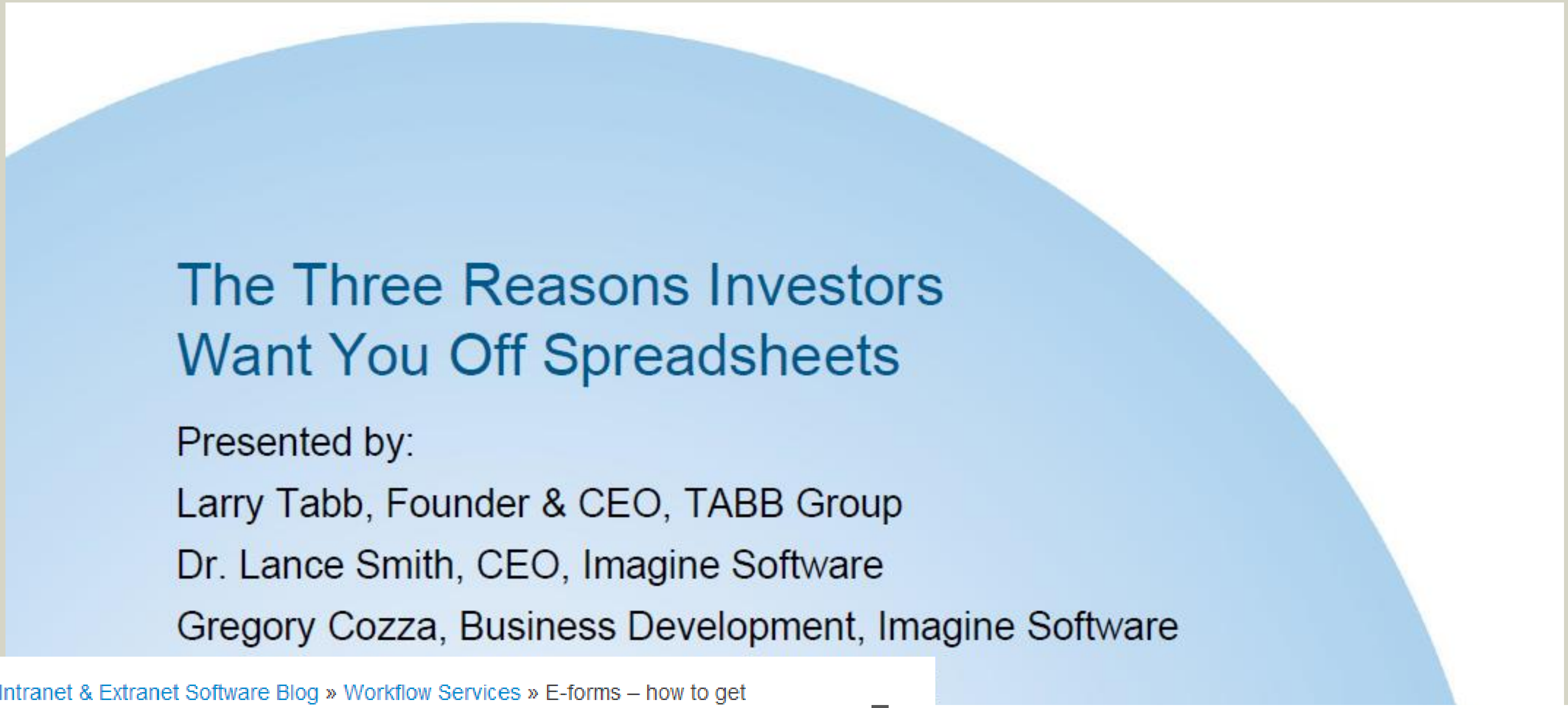
Issues:

from the Web

- Source: MARKETO BLOG
- 1) **Spreadsheet data can be erroneous when entered or changed.** Now, you can argue that this is true of all applications that involve data entry, but spreadsheets are particularly prone to this problem because the inevitable errors are hard to identify and trace. Controlling who and when changes can be made is very problematic as well.
- 2) **Spreadsheet consolidations and aggregations are not for the weak of heart.** Let's consider a marketing budget developed with a spreadsheet where marketing budget and spend by marketing communications, lead generation, event marketing and web marketing are broken out into tabs and allocated to five lines of business or geographies. That's 20 individual spreadsheet matrices that must be aggregated every time a change is made or a current report requested. One change to any of the cells requires re-aggregation of all the matrices. Yes, you can program, emphasis on the word *program*, macros to automate the process, but honestly, it hurts just thinking about it.
- 3) **Spreadsheets are not scalable.** I have seen marketing budgets that encompass hundreds of marketing programs, multiple lines of business or geographies across multiple product lines. If you are tracking and rolling up all marketing spend transactions, you can easily get into tens of thousands of lines of spreadsheet rows making reporting and consolidations slow and difficult.
- 4) **Spreadsheets are in reality not that flexible.** Using the same scenario cited in the consolidations and scalability examples described above, imagine the impact of a decision to add a column, track a new element, add a new category of spending or a new line of business. The impact ripples throughout your spreadsheet based marketing budget. Also, watch out for data entry errors when these changes happen too!!!
- 5) **Spreadsheets are problematic to inspect and audit.** We've all seen it. Your spreadsheet doesn't cross foot or total up to what's in the submitted budget or spending report. Consider a budget that was fine on Monday but is now not footing on Tuesday. Who changed it and where was the change made. What's your audit trail?

Issues:

from the Web



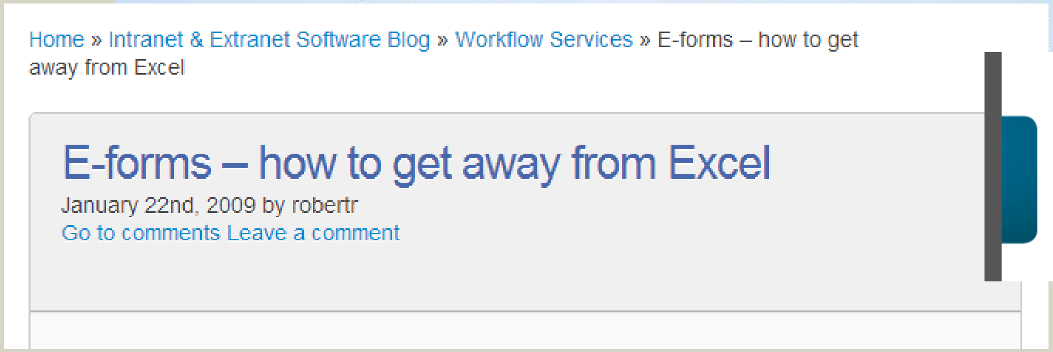
The Three Reasons Investors Want You Off Spreadsheets

Presented by:

Larry Tabb, Founder & CEO, TABB Group

Dr. Lance Smith, CEO, Imagine Software

Gregory Cozza, Business Development, Imagine Software

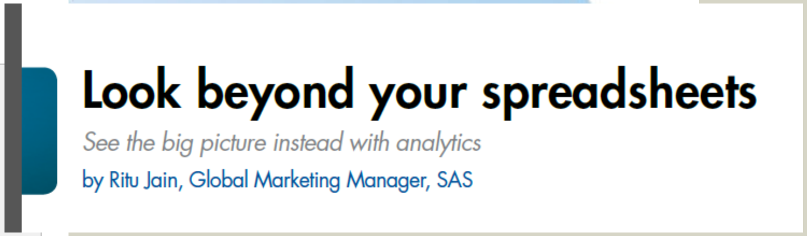


Home » Intranet & Extranet Software Blog » Workflow Services » E-forms – how to get away from Excel

E-forms – how to get away from Excel

January 22nd, 2009 by robertr

Go to comments Leave a comment



Look beyond your spreadsheets

See the big picture instead with analytics

by Ritu Jain, Global Marketing Manager, SAS

Let’s take a different perspective

- **Programming language design perspective:**
  - Data Representation
    - Which objects can be created?
    - Which ones can be named?
    - What kind of type system is implied?
    - How can users combine existing constructs?
  - Functional composition
    - How can users define new functions?
- **UI software design perspective:**
  - How can users present their model?
  - How much control they have over the layout?
  - How easy is to adapt to model growth and changes?

P1: overly simplistic state modeling

- Current Paradigm
  - State modeled as a set of named worksheets
  - Each worksheet is a 2D grid
  - Cells contain formulas or values
  - Cells and ranges are designated by coordinates
  - One-type type system: every cell is a variant
- Naming of cells (and ranges) is a difficult matter
- Resulting nightmares
  - Hard to read, understand, and validate
  - Easy to break

P2:
lack of functional abstraction

- Functions are predefined
  - can create new ones only via VBA, XLL, …
  - result: excessive reliance on copy and paste
- Only first order functions are supported
  - as a result, functions like sumif resort to hacks

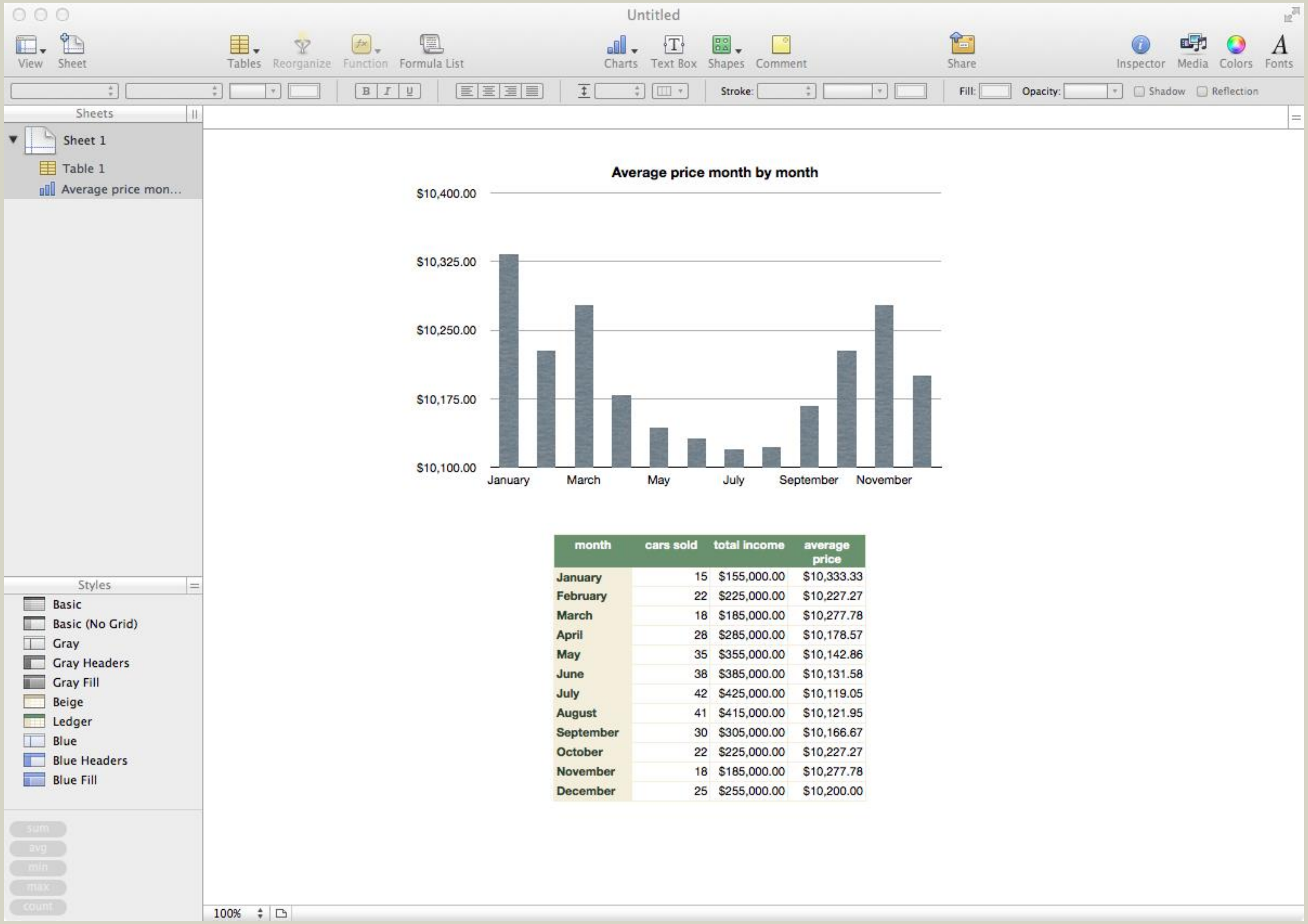
P3: model – view entanglement

- Worksheets have dual personality
  - They are the single data modeling construct …
    - … and the main mechanism to define named variables
    - Most users feel it's overkill to define a worksheet per variable
  - However: Worksheets ⇔ Tabs
  - … and tabs are visualization mechanism!

- Organizing calculations is hard
  - Typical scenario: moving intermediate calculations out of the way
  - Best (modular) solution is to move them to separate worksheets
  - Sure, you can also hide rows and/or columns …
    - … but why should you work so hard to achieve this decluttering goal?

- Typical nightmare:
  - Rearranging the layout to make space for additional calculations
  - Making sure that different sheets share basic parameters
  - … and visualizing said parameters in different sheets

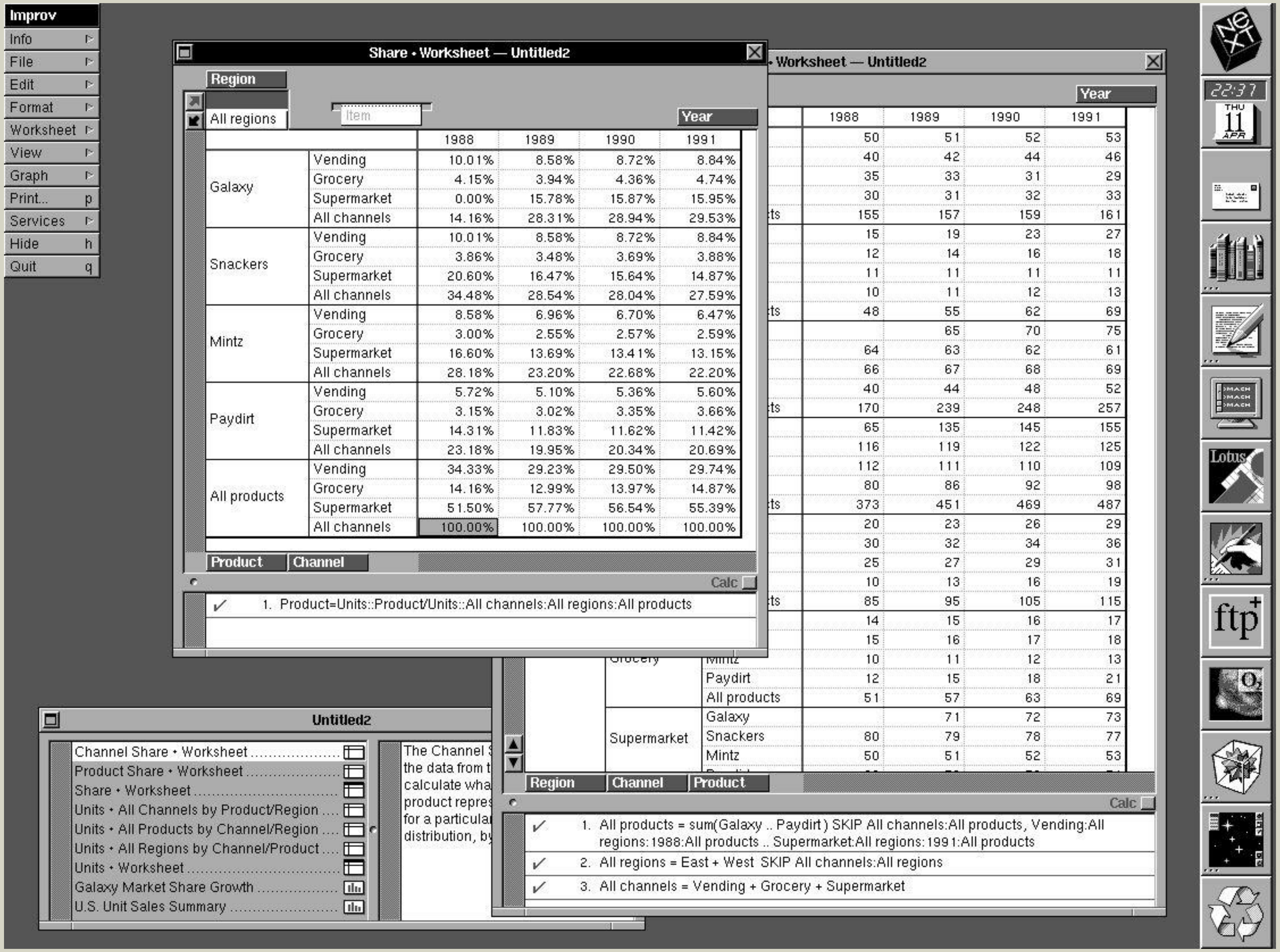
Partial Exceptions: Numbers



month	cars sold	total income	average price
January	15	$155,000.00	$10,333.33
February	22	$225,000.00	$10,227.27
March	18	$185,000.00	$10,277.78
April	28	$285,000.00	$10,178.57
May	35	$355,000.00	$10,142.86
June	38	$385,000.00	$10,131.58
July	42	$425,000.00	$10,119.05
August	41	$415,000.00	$10,121.95
September	30	$305,000.00	$10,166.67
October	22	$225,000.00	$10,227.27
November	18	$185,000.00	$10,277.78
December	25	$255,000.00	$10,200.00

https://en.wikipedia.org/wiki/File:NumbersIcon.png

Partial Exceptions: Lotus Improv



Share • Worksheet — Untitled2

Region: All regions

Item

Product	Channel	1988	1989	1990	1991
Galaxy	Vending	10.01%	8.58%	8.72%	8.84%
	Grocery	4.15%	3.94%	4.36%	4.74%
	Supermarket	0.00%	15.78%	15.87%	15.95%
	All channels	14.16%	28.31%	28.94%	29.53%
Snackers	Vending	10.01%	8.58%	8.72%	8.84%
	Grocery	3.86%	3.48%	3.69%	3.88%
	Supermarket	20.60%	16.47%	15.64%	14.87%
	All channels	34.48%	28.54%	28.04%	27.59%
Mintz	Vending	8.58%	6.96%	6.70%	6.47%
	Grocery	3.00%	2.55%	2.57%	2.59%
	Supermarket	16.60%	13.69%	13.41%	13.15%
	All channels	28.18%	23.20%	22.68%	22.20%
Paydirt	Vending	5.72%	5.10%	5.36%	5.60%
	Grocery	3.15%	3.02%	3.35%	3.66%
	Supermarket	14.31%	11.83%	11.62%	11.42%
	All channels	23.18%	19.95%	20.34%	20.69%
All products	Vending	34.33%	29.23%	29.50%	29.74%
	Grocery	14.16%	12.99%	13.97%	14.87%
	Supermarket	51.50%	57.77%	56.54%	55.39%
	All channels	100.00%	100.00%	100.00%	100.00%

Calc

✓ 1. Product=Units::Product/Units::All channels:All regions:All products

• Worksheet — Untitled2

Region	Channel	Product	1988	1989	1990	1991
			50	51	52	53
			40	42	44	46
			35	33	31	29
			30	31	32	33
		ts	155	157	159	161
			15	19	23	27
			12	14	16	18
			11	11	11	11
			10	11	12	13
		ts	48	55	62	69
				65	70	75
			64	63	62	61
			66	67	68	69
			40	44	48	52
		ts	170	239	248	257
			65	135	145	155
			116	119	122	125
			112	111	110	109
			80	86	92	98
		ts	373	451	469	487
			20	23	26	29
			30	32	34	36
			25	27	29	31
			10	13	16	19
		ts	85	95	105	115
			14	15	16	17
			15	16	17	18
	Grocery	Mintz	10	11	12	13
		Paydirt	12	15	18	21
		All products	51	57	63	69
	Supermarket	Galaxy		71	72	73
		Snackers	80	79	78	77
		Mintz	50	51	52	53

Calc

✓ 1. All products = sum(Galaxy .. Paydirt) SKIP All channels:All products, Vending:All regions:1988:All products .. Supermarket:All regions:1991:All products

✓ 2. All regions = East + West SKIP All channels:All regions

✓ 3. All channels = Vending + Grocery + Supermarket

Untitled2

- Channel Share • Worksheet
- Product Share • Worksheet
- Share • Worksheet
- Units • All Channels by Product/Region
- Units • All Products by Channel/Region
- Units • All Regions by Channel/Product
- Units • Worksheet
- Galaxy Market Share Growth
- U.S. Unit Sales Summary

The Channel S
the data from t
calculate wha
product repres
for a particula
distribution, by

Problems: summary

- Deficient state (data) modeling constructs
  - All models must be represented as a set of 2D grids …
  - … with variables squeezed in a sea of (usually nameless) cells
  - All variables (cells) are of variant type
- Lack of functional abstraction
  - Can't define new functions to abstract repetitive formulas
    - … except by going outside of the pure spreadsheet realm
  - No higher order functions
- Entanglement of model and presentation
  - Sharing parameters across worksheets is error prone
  - Rearranging sheet layouts is a frequent cause of bugs
  - Organizing calculations is unnecessarily hard
- Other issues
  - Lack of support for stateful computations

Problems => Reactions => Symptoms

- Coping mechanisms:
  - copy-and-paste / cut-and-paste
  - naming of cells and ranges
  - multiple worksheets
  - Macros and VBA
- Consequences:
  - Spreadsheets are error prone
  - Hard to inspect and validate
  - Easy to break
  - Not that flexible
  - Not powerful enough for serious computing
  - … see previous laundry list of issues

Smart Sheets Manifesto

- Spreadsheets must
  - Support richer data structures
    - types and orthogonal data composition
    - static data typing (with “optionality” via variant)
  - Provide unfettered functional abstraction
    - ability to define new functions combining existing ones
    - allow higher order functions
  - Separate model from presentation
    - allows for multiple views of the computation model
- Let’s aim higher
  - Why trying to be “just like Excel” …
  - … when we can be better than Excel?

XXI Century Type System

- Scalars
  - boolean, integer, floating point, …
- N-Dimensional Arrays
  - Element selection via subscript
  - Range subsetting along any dimension
- Product types
- Functions

Functional Abstraction

- It should be possible to
  - Assign functions to variables
  - Pass functions as arguments to other functions
  - Return functions as results

- Commonly used higher order functions (built-in)
  - fmap
  - fold
  - ...

Model – View Separation

Better illustrated with a show and tell

DEMO TIME!

Conclusions

- A simple, but not overly simplistic, type system promotes understanding and increases users' ability to maintain correctness
- Functional abstraction is a must
- A computation model must stand on its own, entirely separated from its presentation
- Presentation is a pillar of usability
  - The visual, intuitive, ease of use of spreadsheets should not, and need not, be lost as a consequence of supporting functional abstraction and a richer type system.
  - The pre-paid learning cost need not be thrown out

About Us

- Lakebolt Research
- www.lakebolt.com