



A FrameWork for Financial Modeling

Ralph E. Johnson
University of Illinois at Urbana-Champaign
1304 W. Springfield Ave
Urbana, IL 61801
johnson@cs.uiuc.edu
<http://st-www.cs.uiuc.edu/users/johnson>

with help from Joe Yoder

Sponsored by Caterpillar Inc. through the National Center for
Supercomputer Applications

Copyright 1997 by Ralph E. Johnson

1



Goals

- ◆ Case study of developing a framework
- ◆ Case study of using design patterns
- ◆ Learn a framework for financial modeling

Copyright 1997 by Ralph E. Johnson

2



Overview

- ◆ What is a Financial Model?
- ◆ How we developed our framework.
- ◆ The design of our framework.
- ◆ Patterns in our framework.

Copyright 1997 by Ralph E. Johnson

3



What is a Financial Model?

- ◆ reports
- ◆ answer “why”
- ◆ correct errors, enter budget
- ◆ depends on database
- ◆ ensure security

Copyright 1997 by Ralph E. Johnson

4



Answering Why

- ◆ answers questions about finances
 - profit, return on assets
 - detailed costs
 - compare actual, budget, predicted
- ◆ high-level and detailed
- ◆ fixed reports and ad-hoc queries

Copyright 1997 by Ralph E. Johnson

5



What is a Financial Model?

- ◆ business logic is equations
 - $\text{variable margin} = \text{net sales} - \text{variable cost}$
 - $\text{net sales} = \text{gross sales} - \text{warranty}$
 - $\text{gross sales} = \text{sum } \textit{sales} \text{ column from } \textit{sales_and_transfer} \text{ table}$
- ◆ User interface just as important

Copyright 1997 by Ralph E. Johnson

6

Warning!
All numbers
are fake.

Top Level Dupont Model

Copyright 1997 by Ralph E. Johnson

7

Inventories Drilldown "Show calculation for Value"

Inventories				
	Budget	Actual	Profit +/-	% Change
Prime Products	\$3,273	\$80,857	\$77,585	2370.80%
Production Stores	\$26,000	\$26,525	\$525	2.02%
INVENTORIES	\$29,273	\$107,382	\$78,110	266.84%

From: January 1996 To: May 1996

Copyright 1997 by Ralph E. Johnson

8

Summary Report

Vehicles by marketing company.

The screenshot shows a software window titled "Vehicle Sales Report" with a menu bar (File, Window, Help) and a toolbar. Below the menu is a summary table with columns: Family Type, Model Number, Total, HSD, GAO, CAP, OSA, and OCL. A "Vehicle Sales Specification" dialog box is overlaid on top, showing a grid of checkboxes for various vehicle models and options, organized into columns: Mktg Div, HEX, LWL, MWL, OTH, and Mktg CD. The dialog has "All None", "OK", and "Cancel" buttons. The status bar at the bottom indicates "From: January, 1995" and "To: December, 1995".

Copyright 1997 by Ralph E. Johnson

9

Detailed Transactions

Inspect and edit the individual transactions.

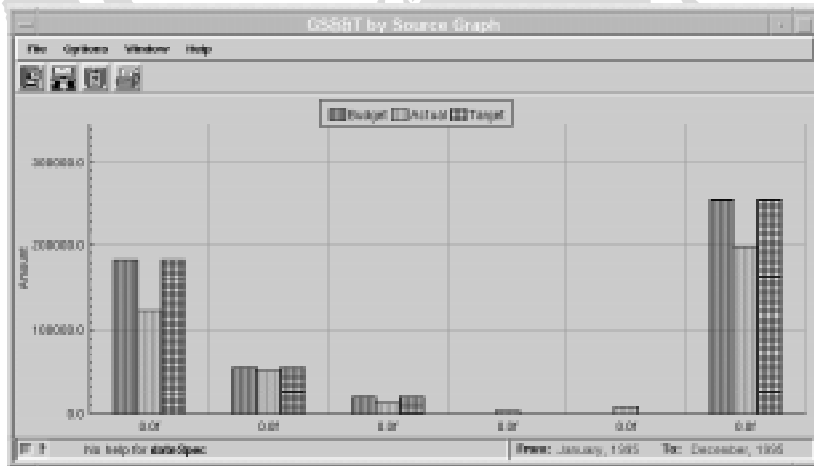
The screenshot shows a software window titled "Query Based On: Dep Expenses Actual" with a menu bar (File, Tools, Window, Help) and a toolbar. Below the menu is a toolbar with buttons: Delete, Accept, Cancel, Commit, Cancel All, and Changes. To the right of these buttons are three small tables: "Selection" (1), "Displayed" (42), and "Total" (16210). The main area contains a data table with the following columns: Date, Family, Section, Account, Seq Key, PCOS Actual, and RD COST. The status bar at the bottom indicates "From: January 1996" and "To: May 1996".

Date	Family	Section	Account	Seq Key	PCOS Actual	RD COST
Jan-96	OTH	05050	605	05	0	0
Jan-96	HEX	05050	606	05	0	0
Jan-96	LWL	05050	606	05	0	0
Jan-96	MWL	05050	606	05	0	0
Jan-96	SKD	05050	606	05	0	0
Jan-96	OTH	05050	607	04	0	0

Copyright 1997 by Ralph E. Johnson

10

Graphs



Copyright 1997 by Ralph E. Johnson

11

Summary of Reports

- ◆ Top level (Dupont or P&L)
- ◆ Drill down (ReportModel)
- ◆ Summary report
- ◆ Detailed transactions
- ◆ Graphs

Copyright 1997 by Ralph E. Johnson

12

Patterns for Developing Frameworks



- 1) Three Examples
- 2) White-box Framework
- 3) Component Library
 - Build applications and add components to library
- 4) Hot Spots
 - Separate Changeable from Stable Code
 - Design Patterns

Copyright 1997 by Ralph E. Johnson

13

Patterns for Developing Frameworks



- 5) Pluggable Objects
- 6) Fine-grained Objects
- 7) Black-box Framework
- 8) Visual Builder
- 9) Language Tools

[http://st-
www.cs.uiuc.edu/users/droberts/evolve.htm](http://st-www.cs.uiuc.edu/users/droberts/evolve.htm)

1

Copyright 1997 by Ralph E. Johnson

14

Three Examples

- ◆ Models for three business units
- ◆ Seemed completely different at first.
- ◆ Only one was fully implemented

Copyright 1997 by Ralph E. Johnson

15

White-box Framework

Five kinds of ApplicationsModel, with lots of subclasses

- ◆ DupontModel
- ◆ ReportModel
- ◆ DetailedModel
- ◆ SummaryModel
- ◆ GraphModel

Copyright 1997 by Ralph E. Johnson

16

User Interface Frameworks

- ◆ *DuPontModel* -
- ◆ *ReportModel* - Builds a spreadsheet interface using values and GUI descriptions from *ReportValues*.
- ◆ *SummaryReports*
- ◆ *DetailedWindows* - Edit and view individual transactions
- ◆ *GraphReports*

Copyright 1997 by Ralph E. Johnson

17

White-box Framework

- ◆ New window = new subclass
- ◆ Subclass has methods for
 - reading database
 - computing values
 - stuffing them in GUI
- ◆ Initialization registers with dependents

Copyright 1997 by Ralph E. Johnson

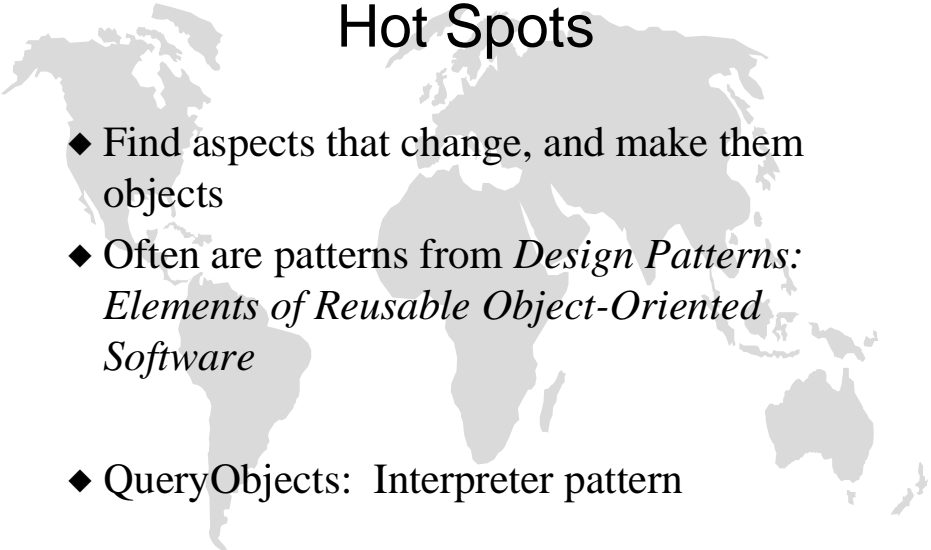
18



Component Library

- ◆ First, just abstract superclasses
- ◆ Second, query objects
- ◆ Third, GUI objects

Copyright 1997 by Ralph E. Johnson 19



Hot Spots

- ◆ Find aspects that change, and make them objects
- ◆ Often are patterns from *Design Patterns: Elements of Reusable Object-Oriented Software*
- ◆ QueryObjects: Interpreter pattern

Copyright 1997 by Ralph E. Johnson 20

Interpreter Pattern

- ◆ Need to represent SQL to manipulate query:

```
SELECT SUM(sales) FROM sales_and_transfer
WHERE family='MWL' AND date < '1/1/96'
AND date > '1/1/97'
```

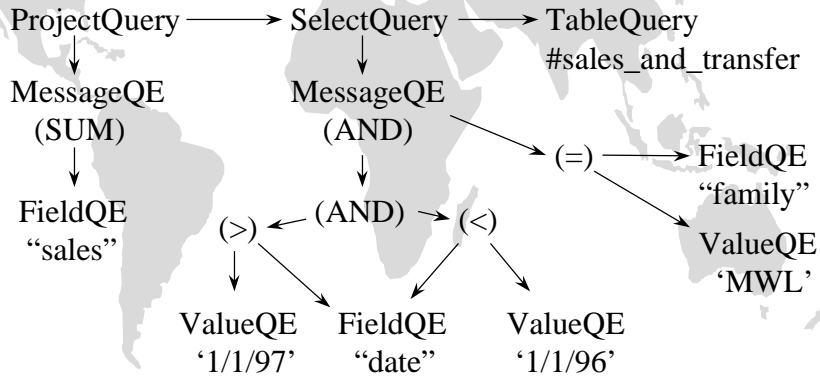
- ◆ Problem: how do you represent a simple language?

Interpreter Pattern

- 1) make a class hierarchy that represents nodes in abstract syntax tree (SELECT, AND, <, tables, field names)
- 2) define methods to construct and manipulate tree
- 3) define method to compute value of query (this is the “interpreter”)

Instance Hierarchy

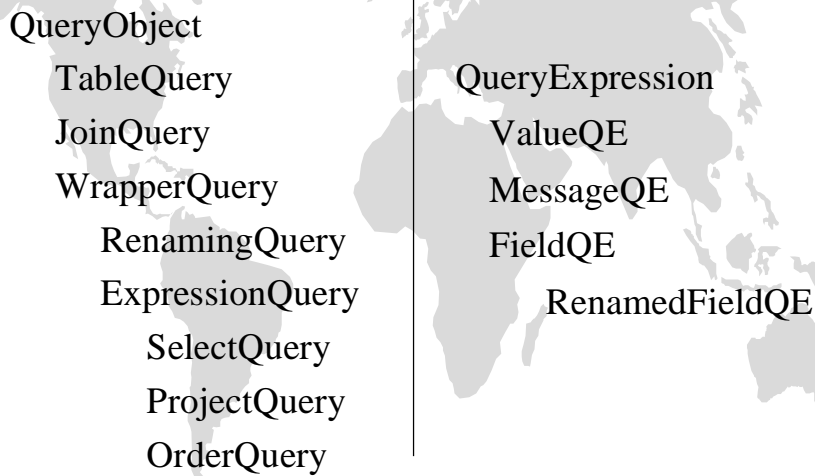
```
SELECT SUM(sales) FROM sales_and_transfer
WHERE family='MWL' AND date < '1/1/96' AND date > '1/1/97'
```



Copyright 1997 by Ralph E. Johnson

23

QueryObjects



Copyright 1997 by Ralph E. Johnson

24

QueryObject Protocol

- ◆ values - answer collection of tuples
- ◆ fieldNames
- ◆ join: aQueryObject
- ◆ select: aQueryExpression
- ◆ project:, renameColumnsTo:, outerJoin:, groupBy:, orderBy:, asDistinct

Creating a QueryObject

```
salesQ := #sales_and_transfer asQuery.  
dateQ := salesQ select:  
    ((salesQ @@ 'family') = 'MWL') &  
    ((salesQ @@ 'date') > '1/1/96') &  
    ((salesQ @@ 'date') < '1/1/97').  
dateQ project: (dateQ @@ 'sales') Sum
```

QueryExpression Protocol

+, -, <, =, &, |, Sum, Average, Count, ...

Sending one of these messages to a QueryExpression builds a MessageQE with the appropriate operands, and with the message as the operator.

Copyright 1997 by Ralph E. Johnson

27

Leading to Black-box

- ◆ Component Library
- ◆ Hot Spots
- ◆ Pluggable Objects
- ◆ Fine-grained Objects
- ◆ Black-box Frameworks

Copyright 1997 by Ralph E. Johnson

28

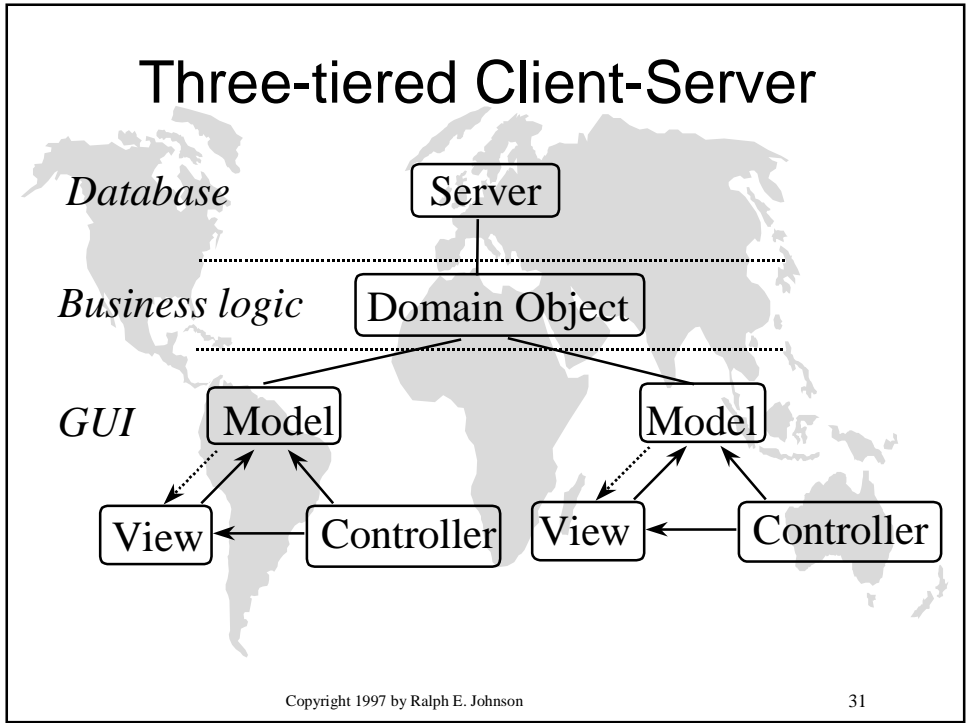
First Design

- ◆ Class hierarchy of ReportModels, ReportModel creates QueryObjects.
- ◆ Improvement: separate logic and GUI
- ◆ Two hierarchies: ReportModel and ReportValues.
 - Result: twice the classes, some reuse

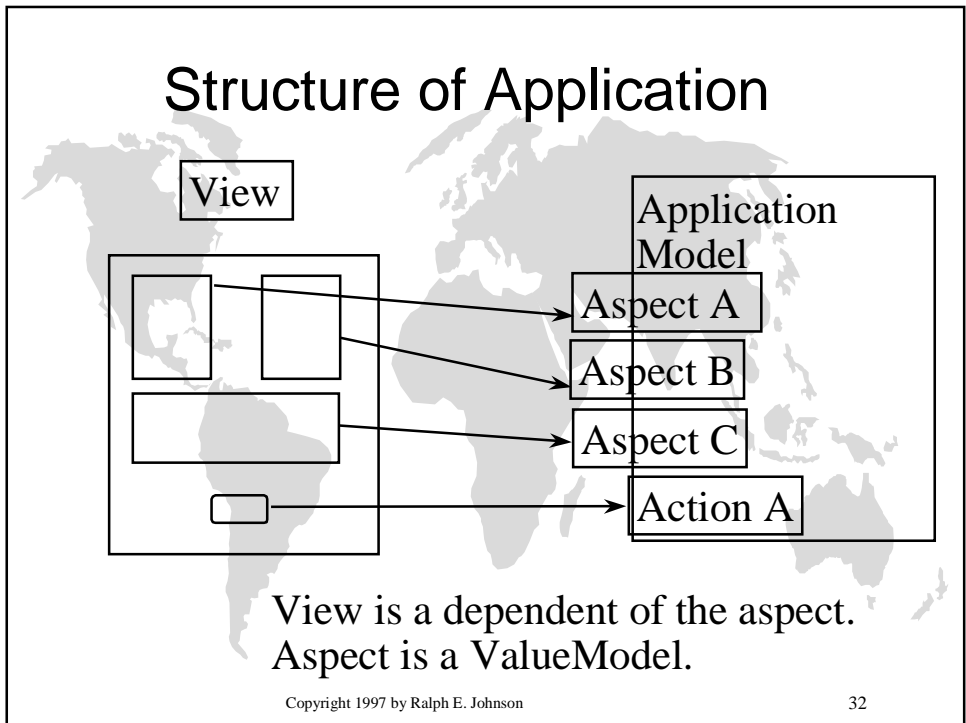
First Separation

- | | |
|-------------------|--------------------|
| ◆ ReportModel | ◆ ReportValues |
| ◆ SalesModel | – SalesValues |
| ◆ InventoryModel | – InventoryValues |
| ◆ ... | – ... |
| Uses QueryObjects | Makes QueryObjects |

Three-tiered Client-Server



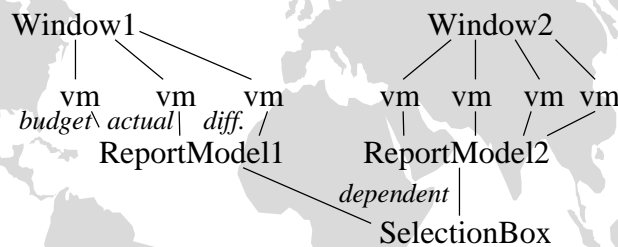
Structure of Application



ValueModel

- ◆ Example of Observer pattern
- ◆ View (observer) registers with ValueModel (subject) and is notified when it changes.
- ◆ ValueModel protocol
 - value, value:
 - addDependent:, removeDependent
- ◆ Observer protocol
 - update:

Using ValueModels



budget value: (query1 values first first).
actual value: (query 2 values first first).
difference value: budget value - actual value

A query returns a collection of tuples.

Alternative Solutions

- ◆ ReportModel depends on SelectionBox.
 - updates for too many changes
- ◆ ReportModel depends on ValueModels from SelectionBox used in QueryObject
 - hard to manage dependencies
- ◆ ReportModel depends on QueryObject, QueryObject depends on ValueModels from SelectionBox

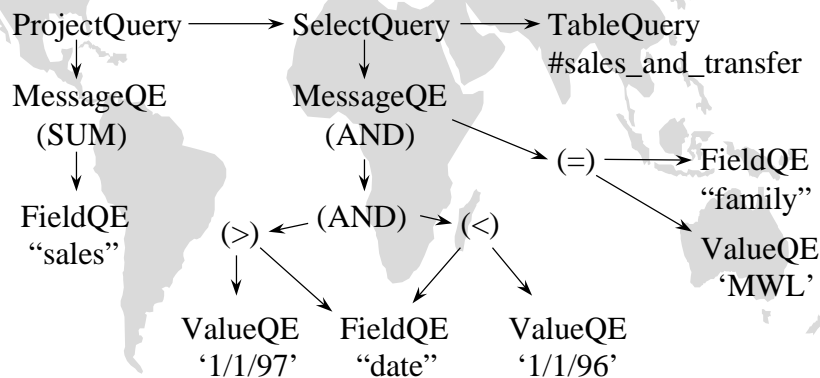
Copyright 1997 by Ralph E. Johnson

35

Observer and QueryObjects

Let ValueQE refer to a ValueModel.

Let each QueryObject observe its components.



Copyright 1997 by Ralph E. Johnson

36

Old way - route change through report

budget value: (query1 values first first).
actual value: (query 2 values first first).
difference value: budget value - actual value

Update

New way - route change directly to ValueModel

budget := QueryHolder on: query1
actual := QueryHolder on: query2
difference := budget - actual

Initialization

Requires:

QueryHolder - adapts QueryObject to ValueModel
ValueModel understands +, -, *, /, etc

QueryHolder

Adaptor pattern - subclass of ValueModel that
lets QueryObject act like ValueModel.

instance variables: query, values

query: aQuery

query := aQuery.

aQuery addDependent: self

QueryHolder

update
values := aQuery values
self changed

value
^values first first

Copyright 1997 by Ralph E. Johnson

39

Arithmetic on ValueModels

ValueModel implements arithmetic by
creating ValueModels that compute
function.

+ anObject

^BlockValue

on: [:a :b | a value + b value]

with: (Array with: self with: anObject)

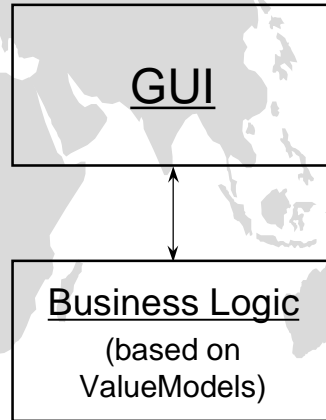
Copyright 1997 by Ralph E. Johnson

40

Result of Refactoring

Reuse GUIs, change ValueModels.

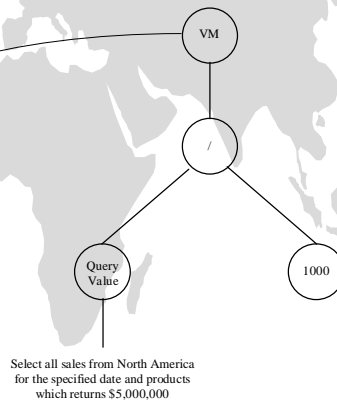
Hard part is creating ValueModels and connecting them to GUI.



Typical Values in a Report

budget	actual	variance
	\$ 5,000	

Thousands of Dollars



Select all sales from North America for the specified date and products which returns \$5,000,000



Business logic is equations expressed with ValueModel and QueryObjects

- ◆ Values = functions of other values
- ◆ Values = queries from the database

variable margin = net sales - variable cost

net sales = gross sales - warrantee

gross sales = sum *sales* column from
sales_and_transfer table



Problems

- ◆ How do we go from one report to the next?
- ◆ How do we connect report to business model?
- ◆ Must define business model flexibly
- ◆ Must define GUI flexibly

Specifications

- ◆ A ReportSpec
 - has name
 - has parameters
 - has menus, which name other reports
- ◆ DetailedReportSpec and SummaryReportSpec are parameterized with QueryObjects.
- ◆ GraphReportSpec is parameterized with ValueModels

ReportValues

Many tables.
 Many columns
 Each table has a
 sequence of
 valueModels
 Total at end.

R & D			
	Budget	Actual	Profit +/-
Internal R&D	\$1,798	\$5,342	(\$3,544)
Inbound R&D	\$1,225	\$9,293	(\$8,068)
Outbound R&D	\$52	(\$1,281)	\$1,229
Total R&D	\$2,975	\$13,344	(\$10,369)
Office Costs			
Commercial & General	\$215	\$815	(\$600)
Planning	\$2,568	\$2,755	(\$187)
Internal Services	\$5,187	\$7,898	(\$2,711)
Inbound Services	\$757	\$3,529	(\$2,772)
Outbound Services	\$82	\$492	(\$410)
Total Office Costs	\$8,894	\$17,497	(\$8,603)
Factory Costs			
Depreciation	\$411	\$5,873	(\$5,462)
Occupancy Costs	\$1,488	\$1,898	(\$1,110)
Mach./Equip. Repair	\$1,081	\$1,242	(\$161)
Other Prod. Costs	\$183	\$5,571	(\$5,388)
Total Factory Costs	\$4,163	\$13,584	(\$9,421)
PERIOD COSTS	\$14,738	\$33,425	(\$18,687)

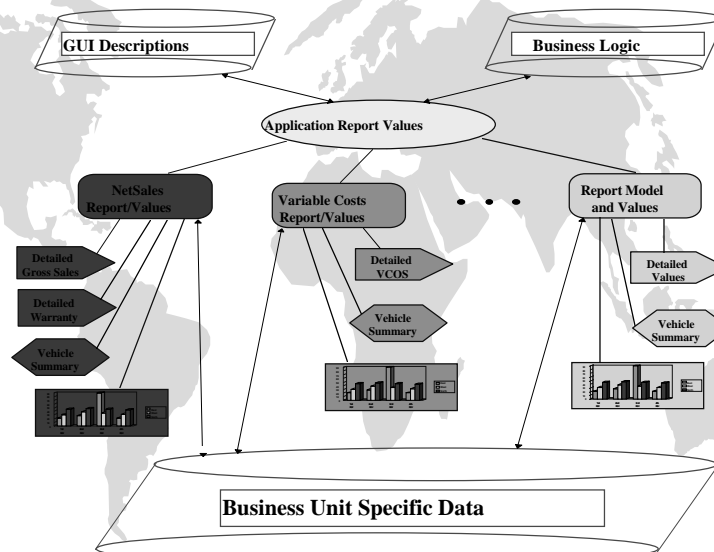
Solution

- ◆ ReportValues responsible for
 - knowing values
 - knowing how to compute values
 - knowing how to display more detail (drill-down) on values
- ◆ Top level starts up ReportValues which starts up next.

Copyright 1997 by Ralph E. Johnson

47

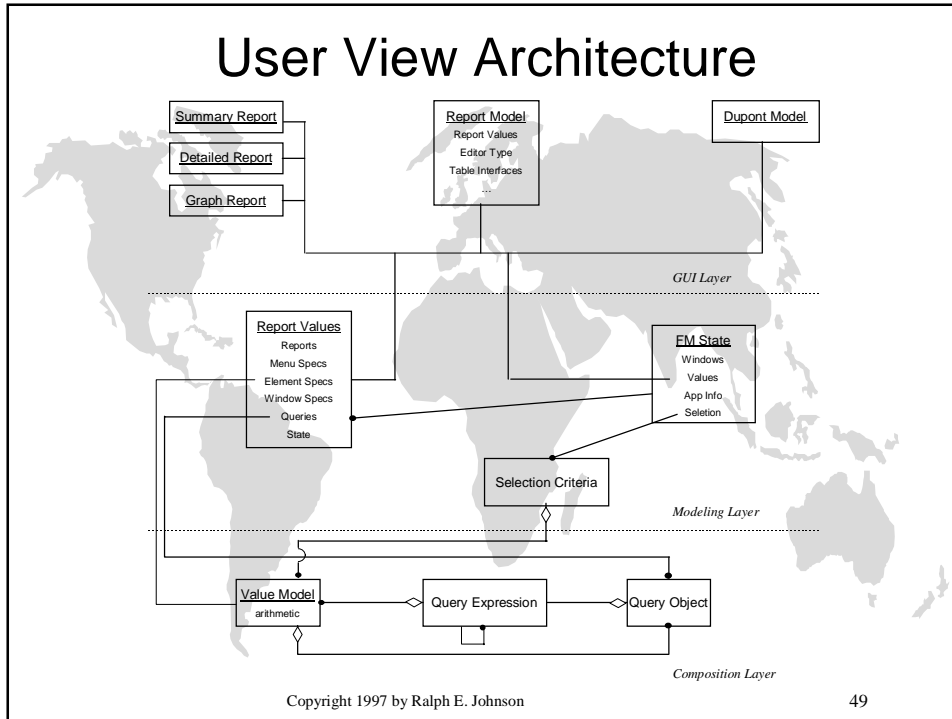
Report Values



Copyright 1997 by Ralph E. Johnson

48

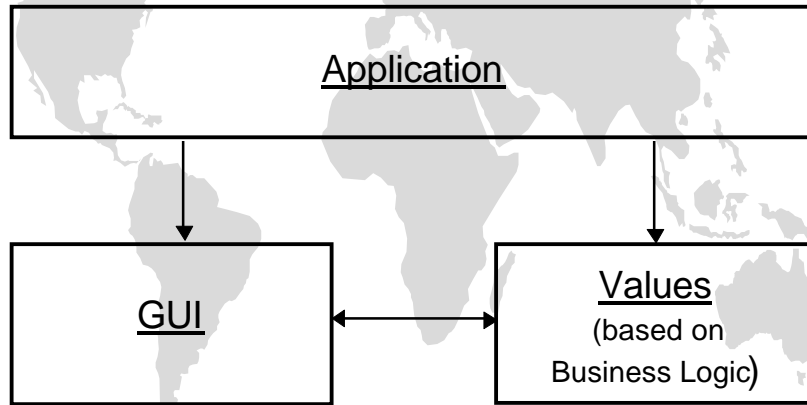
User View Architecture



ReportValue protocol

- ◆ budget, actual - ValueModels
- ◆ openEditor - open “drill down”
 - specify spreadsheets, ValueModels to go in the spreadsheets, menus, reports on menus
- ◆ openWith: aSymbol - opens named report

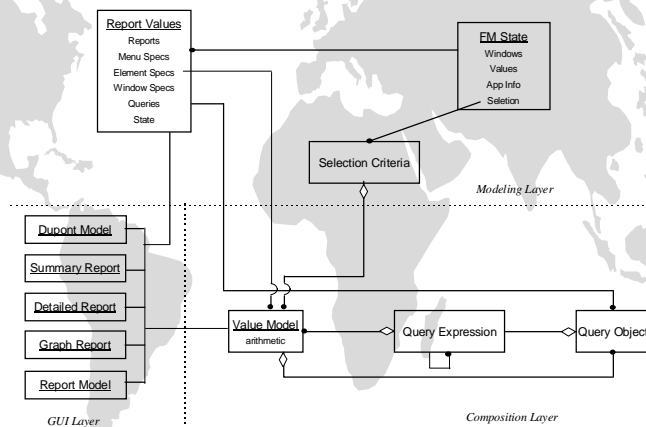
Layered Architecture



Copyright 1997 by Ralph E. Johnson

51

Black-box Framework



Copyright 1997 by Ralph E. Johnson

52

Visual Builder

- ◆ Make a GUI to define Specs.
- ◆ This GUI is a language for defining financial models.

Copyright 1997 by Ralph E. Johnson

53

Builders

- ◆ Equations in a Report Value
 - expressions
 - queries
- ◆ GUIs
 - Report Value (Drill down)
 - Graphs - specify business logic, labels
 - Detailed - specify query, labels, editing
 - Summaries - specify query, grouping, columns to sum and calculate
- ◆ Selection

Copyright 1997 by Ralph E. Johnson

54

Language Tools

- ◆ Languages need debuggers, profilers, version control, etc.
- ◆ So far only built a testing tool.

Copyright 1997 by Ralph E. Johnson

55

Summary of Architecture

- ◆ Builders
- ◆ ReportValues, Selection Criterion, FMState
- ◆ GUI frameworks
- ◆ ValueModel, QueryObject

Copyright 1997 by Ralph E. Johnson

56

Summary of Architecture

- ◆ business model is not object-oriented, just a bunch of equations
- ◆ object model is the language for specifying business model, not the business model

Copyright 1997 by Ralph E. Johnson

57

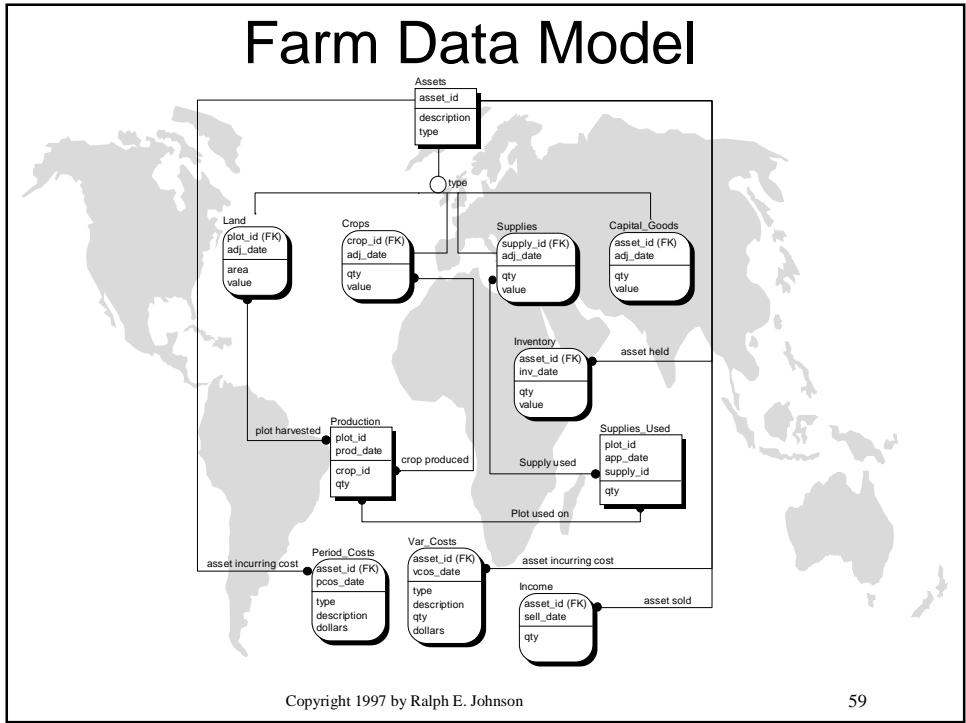
Data Model

- ◆ Application Specific
 - holds “real data”
 - changes with every business model
- ◆ Generic
 - specifies business logic and GUI
 - never changes

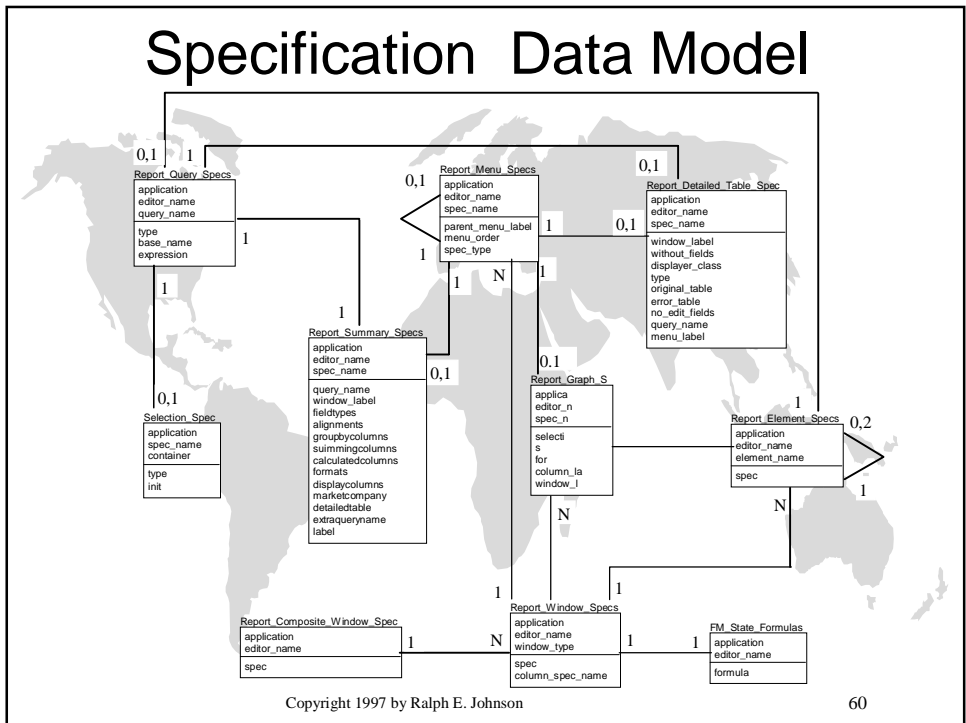
Copyright 1997 by Ralph E. Johnson

58

Farm Data Model



Specification Data Model



Other Features

- ◆ Any window can print itself
- ◆ Automated testing support
- ◆ Security for editing and/or viewing the data; configured by administrators.

Copyright 1997 by Ralph E. Johnson

61

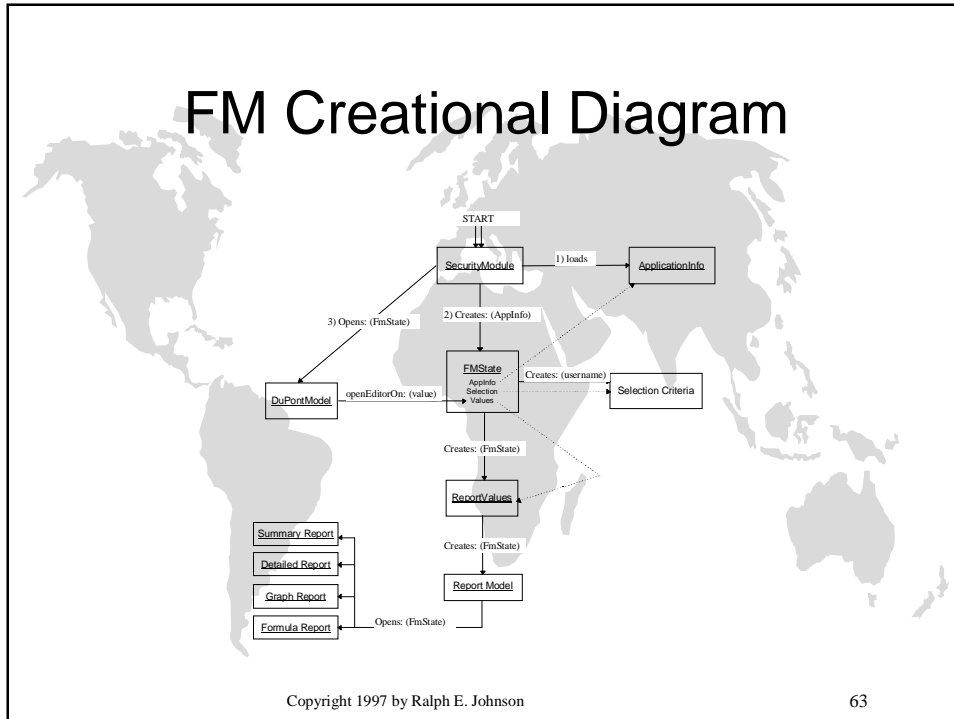
Security Requirements

- ◆ Control passwords
- ◆ Control login
- ◆ Users have roles
- ◆ Role can only view a specified list of products.
- ◆ Role can only edit a subset of the specified list of products.
- ◆ All security features can be controlled by administrators

Copyright 1997 by Ralph E. Johnson

62

FM Creational Diagram



How to Develop a New Financial Application

- ◆ Analyze business unit
- ◆ Build business-unit data model
- ◆ Specify GUI and business logic
- ◆ Install and Test

Analyze Business Unit

- ◆ Questions to ask a new business unit
 - Values to be calculated (netsales, vcos, pcos, ...)
 - User interface
 - ◆ top level
 - ◆ Drill Downs (summary and detailed)
 - ◆ Graphs of values
 - Error-Correction/Analysis modules

Copyright 1997 by Ralph E. Johnson

65

Summary

- ◆ We have developed a reusable design for financial applications
- ◆ Domain specific “Visual-Language”
- ◆ Framework emerges by repeatedly refactoring system to eliminate complexity and create flexibility

For more information, see

http://cat.ncsa.uiuc.edu/~yoder/financial_framework/

Copyright 1997 by Ralph E. Johnson

66