

# A Framework for Financial Modeling

Joseph W. Yoder  
University of Illinois at Urbana-Champaign  
1304 W. Springfield Ave  
Urbana, IL 61801  
j-yoder@uiuc.edu  
<http://www.uiuc.edu/ph/www/j-yoder>

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

Copyright 1997 by Joseph W. Yoder & 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 Joseph W. Yoder & Ralph E. Johnson

2

# Overview

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

Copyright 1997 by Joseph W. Yoder & Ralph E. Johnson

3

# Frameworks

- ◆ Interface design and functional factoring constitute the key intellectual content of software and are far more difficult to create or re-create than code.

Peter Deutsch

- ◆ Difference between framework and component library for framework (components get plugged in and are concrete sub-classes that do all the work).

For detailed information about Frameworks attend  
Ralph Johnson's tutorial or see:

<http://st-www.cs.uiuc.edu/users/johnson/>

Copyright 1997 by Joseph W. Yoder & Ralph E. Johnson

4

# Frameworks Encode Domain Knowledge

- ◆ Frameworks solve a particular set of problems.
  - get different points of view
  - explain/defend current design
- ◆ Some frameworks are more technology (horizontal) frameworks verses application domain (vertical) frameworks.
  - Are we solving GUI's, object persistence verses more application domain related such as insurance or manufacturing.

Copyright 1997 by Joseph W. Yoder & Ralph E. Johnson

5

# Frameworks

- ◆ Framework is:
  - reusable design of an application or subsystem
  - represented by a set of abstract classes and the way objects in those classes collaborate.
- ◆ Use framework to build application by:
  - Creating new subclasses
  - Configuring objects together
  - Modifying working examples

Copyright 1997 by Joseph W. Yoder & Ralph E. Johnson

6

## Frameworks

- ◆ Framework prescribes how to decompose a problem.
- ◆ Not just the classes, but the way instances of the classes collaborate.
  - shared invariants that objects must maintain, and how they maintain them
  - framework imposes a collaborative model that you must adapt to.

Copyright 1997 by Joseph W. Yoder & Ralph E. Johnson

7

## Relevant Principles

- ◆ Frameworks are abstractions: people generalize from concrete examples
- ◆ Designing reusable code requires iteration
- ◆ Frameworks encode domain knowledge
- ◆ Customer of framework is application programmer

Copyright 1997 by Joseph W. Yoder & Ralph E. Johnson

8

## Generalize from Concrete Cases

- ◆ People think concretely, not abstractly.
- ◆ Abstractions are found bottom-up, by examining concrete examples.
- ◆ Generalization proceeds by
  - finding things that are given different names but are really the same,
  - parameterizing to eliminate differences,
  - breaking large things into small things so that similar components can be found, and
  - categorizing things that are similar.

Copyright 1997 by Joseph W. Yoder & Ralph E. Johnson

9

## Finding Abstract Classes

- ◆ Abstract classes are discovered by generalizing from concrete classes.
- ◆ To give two classes a common superclass:
  - give them common interface
    - + rename operations so classes use same names
    - + reorder arguments, change types of arguments, etc.
    - + refactor (split or combine) operations
  - if operations have same interface but different implementation, make them abstract
  - if operations have same implementation, move to superclass

Copyright 1997 by Joseph W. Yoder & Ralph E. Johnson

10

# Frameworks Require Iteration

Reusable code requires many iterations.

Basic law of software engineering

If it hasn't been tested, it doesn't work.

Corollary: software that hasn't been reused is not reusable.

Copyright 1997 by Joseph W. Yoder & Ralph E. Johnson

11

# White-box vs. Black-box

**White-box** ←————→ **Black-box**

Customize by subclassing

Emphasize inheritance

Must know internals

Simpler, easier to design

Harder to learn, requires more programming.

Easier to customize because you can overwrite the code

Customize by configuring

Emphasize polymorphism

Must know interfaces

Complex, harder to design

Easier to learn, requires less programming.

Harder to customize because you need to learn how objects collaborate

Copyright 1997 by Joseph W. Yoder & Ralph E. Johnson

12

## What is a Financial Model?

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

Copyright 1997 by Joseph W. Yoder & Ralph E. Johnson

13

## 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 Joseph W. Yoder & Ralph E. Johnson

14

# What is a Financial Model?

- ◆ business logic is equations
  - variable margin = net sales - variable cost
  - net sales = gross sales - warranty
  - gross sales = sum *sales* column from *sales\_and\_transfer* table
- ◆ User interface just as important

Warning!  
All numbers  
are fake.

## Top Level *Dupont Model*

Net Sales & Taxes		Variable Costs		Operating Profit		Net Profit		Accumulated Profit	
\$7,576	100.0%	\$1,105	(14.7%)	\$6,471	(85.3%)	\$1,370	(21.2%)	\$11,310	(158.7%)
\$17,289	228.3%	\$1,238	11.2%	\$16,051	216.5%	\$2,000	145.9%	\$2,000	(17.7%)
\$9,876	130.3%	\$10,276	134.8%	\$2,600	38.6%	\$2,600	191.4%	\$2,600	(23.0%)
\$15,462	204.1%	\$10,276	134.8%	\$5,186	69.1%	\$5,186	383.2%	\$5,186	(46.7%)
\$25,279	333.7%	\$25,279	333.7%	\$1,370	18.5%	\$1,370	103.6%	\$1,370	(12.4%)
\$187,282	2472.0%	\$187,282	2472.0%	\$1,370	18.5%	\$1,370	103.6%	\$1,370	(12.4%)
\$5	0.0%	\$4,286	57.2%	\$7,576	101.3%	\$7,576	565.3%	\$7,576	(67.9%)
\$5	0.0%	\$10,258	135.4%	\$17,289	228.3%	\$17,289	1293.3%	\$17,289	(155.6%)
\$5	0.0%	\$5	0.0%	\$102,833	1362.8%	\$102,833	7724.6%	\$102,833	(917.9%)
\$5	0.0%	\$5	0.0%	\$102,833	1362.8%	\$102,833	7724.6%	\$102,833	(917.9%)

# Inventories Drilldown

## “Show calculation for Value”

	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 Joseph W. Yoder & Ralph E. Johnson

17

# Summary Report

## Vehicles by marketing company.

Entity Type	Model Number	Total	H900	C900	C901H	C901A	C901
H900	343	3,511	3,511				
L900	320	\$17,878,808	\$18,011,493	\$132,142	\$5,263,844	\$753,488	\$144,520
L900	322	\$5,998,833	\$5,933,033				
				3,488	\$173,440	\$220,218	
				1,289	\$1,488,854	\$281,528	
				1,328	\$58,850	\$289,448	
				3,085	\$7,758,728	\$1,510,668	\$144,520
				3,488	\$408,856		
				3,488	\$301,472		
				3,488	\$548,528		
				3,600	\$248,848		

Copyright 1997 by Joseph W. Yoder & Ralph E. Johnson

18

# Detailed Transactions

Inspect and edit the individual transactions.

Query Based On: Dep Expenses Actual

File Tools Window Help

Delete Accept Cancel Commit Cancel All Changes

	Date	Family	Section	Account	Seq Key	PCOS Actual	RD COST
	Jan-96	DTH	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	DTH	05050	607	04	0	0

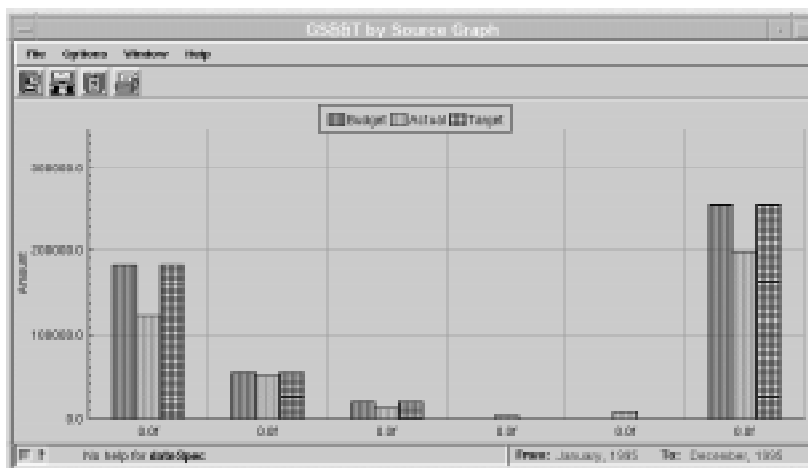
Selection: 1    Displayed: 42    Total: 16210

From: January 1996    To: May 1996

Copyright 1997 by Joseph W. Yoder & Ralph E. Johnson

19

# Graphs



Copyright 1997 by Joseph W. Yoder & Ralph E. Johnson

20

## Summary of Reports

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

## 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

## 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.html>

## Three Examples

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

## White-box Framework

Five kinds of ApplicationsModel, with lots of subclasses

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

## User Interface Frameworks

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

## White-box Framework

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

## Component Library

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

## 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

## 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

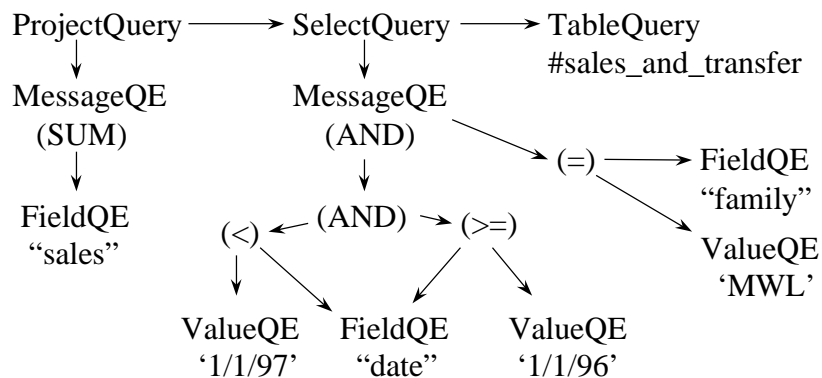
- 1) make a class hierarchy that represents nodes in abstract syntax tree (SELECT, AND, <, tables, field names)
- 2) define methods to construct and manipulate tree
- 3) define method to compute value of query (this is the “interpreter”)

Copyright 1997 by Joseph W. Yoder & Ralph E. Johnson

31

# 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 Joseph W. Yoder & Ralph E. Johnson

32

# QueryObjects

QueryObject

TableQuery

JoinQuery

WrapperQuery

RenamingQuery

ExpressionQuery

SelectQuery

ProjectQuery

OrderQuery

QueryExpression

ValueQE

MessageQE

FieldQE

RenamedFieldQE

# 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.

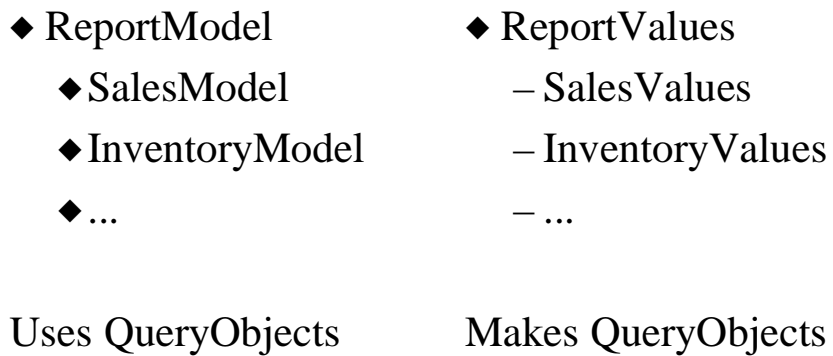
## Leading to Black-box

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

## 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

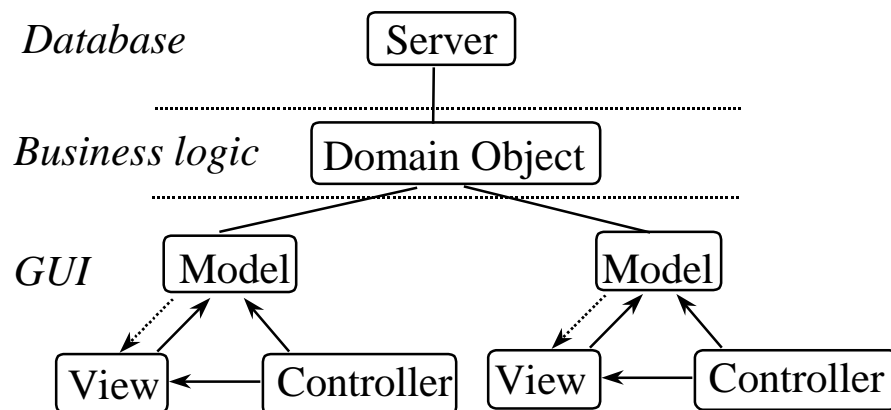
## First Separation



Copyright 1997 by Joseph W. Yoder & Ralph E. Johnson

39

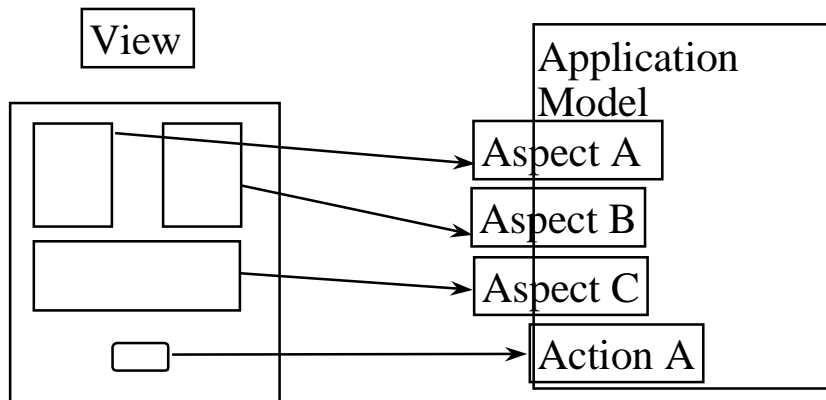
## Three-tiered Client-Server



Copyright 1997 by Joseph W. Yoder & Ralph E. Johnson

40

## Structure of Application



View is a dependent of the aspect.  
Aspect is a ValueModel.

Copyright 1997 by Joseph W. Yoder & Ralph E. Johnson

41

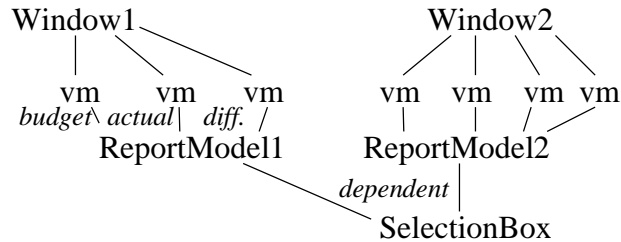
## ValueModel

- ◆ 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:

Copyright 1997 by Joseph W. Yoder & Ralph E. Johnson

42

## 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.*

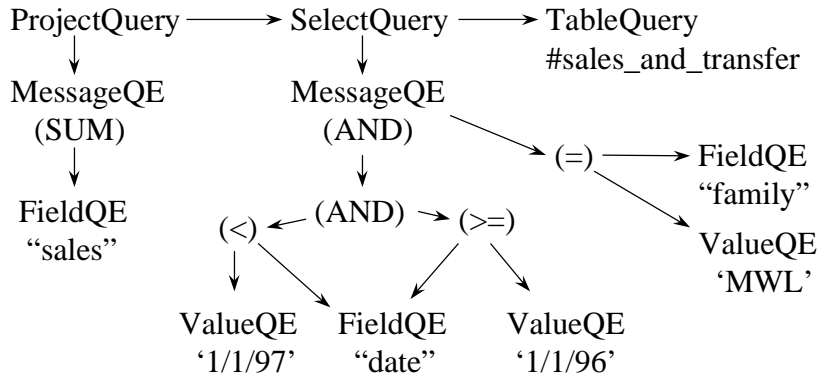
## Alternative Solutions

- ◆ 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

# Observer and QueryObjects

Let ValueQE refer to a ValueModel.

Let each QueryObject observe its components.



Copyright 1997 by Joseph W. Yoder & Ralph E. Johnson

45

## 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

Copyright 1997 by Joseph W. Yoder & Ralph E. Johnson

46

## QueryHolder

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

*instance variables:* query, values

query: aQuery

query := aQuery.

aQuery addDependent: self

Copyright 1997 by Joseph W. Yoder & Ralph E. Johnson

47

## QueryHolder

update

values := aQuery values

self changed

value

^values first first

Copyright 1997 by Joseph W. Yoder & Ralph E. Johnson

48

## Arithmetic on ValueModels

ValueModel implements arithmetic by creating ValueModels that compute function.

+ anObject

^BlockValue

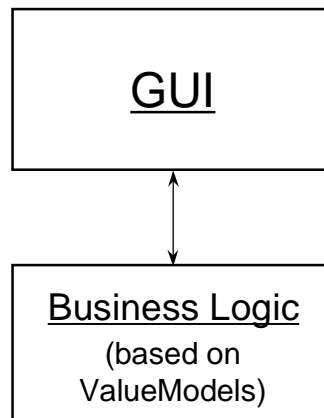
on: [:a :b | a value + b value]

with: (Array with: self with: anObject)

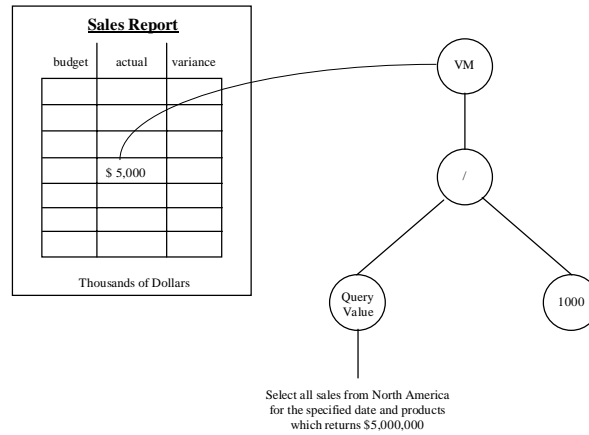
## Result of Refactoring

Reuse GUIs, change ValueModels.

Hard part is creating ValueModels and connecting them to GUI.



# Typical Values in a Report



Copyright 1997 by Joseph W. Yoder & Ralph E. Johnson

51

## 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

Copyright 1997 by Joseph W. Yoder & Ralph E. Johnson

52

## 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.

	Budget	Actual	Profit +/-	% Change
<b>R &amp; D</b>				
Internal R&D	\$1,798	\$5,342	(\$3,544)	(197.40%)
Inbound R&D	\$1,228	\$8,292	(\$7,064)	(574.81%)
Outbound R&D	\$52	(\$1,281)	(\$1,333)	(2563.10%)
Total R&D	\$2,978	\$18,344	(\$15,366)	(5163.59%)
<b>Office Costs</b>				
Commercial & General	\$218	\$815	(\$597)	(273.85%)
Planning	\$2,568	\$2,752	(\$184)	(7.16%)
Internal Services	\$5,182	\$7,898	(\$2,716)	(52.40%)
Inbound Services	\$752	\$3,529	(\$2,777)	(369.11%)
Outbound Services	\$82	\$492	(\$410)	(500.00%)
Total Office Costs	\$7,212	\$18,218	(\$11,006)	(1527.35%)
<b>Factory Costs</b>				
Depreciation	\$411	\$5,873	(\$5,462)	(1328.95%)
Occupancy Costs	\$1,488	\$1,898	(\$410)	(27.55%)
Mach./Equip. Repair	\$1,681	\$1,242	\$439	26.11%
Other Prod. Costs	\$148	\$5,571	(\$5,423)	(3663.51%)
Total Factory Costs	\$4,728	\$13,584	(\$8,856)	(1871.11%)
PERIOD COSTS	\$14,718	\$33,108	(\$18,390)	(1249.31%)

Copyright 1997 by Joseph W. Yoder & Ralph E. Johnson

55

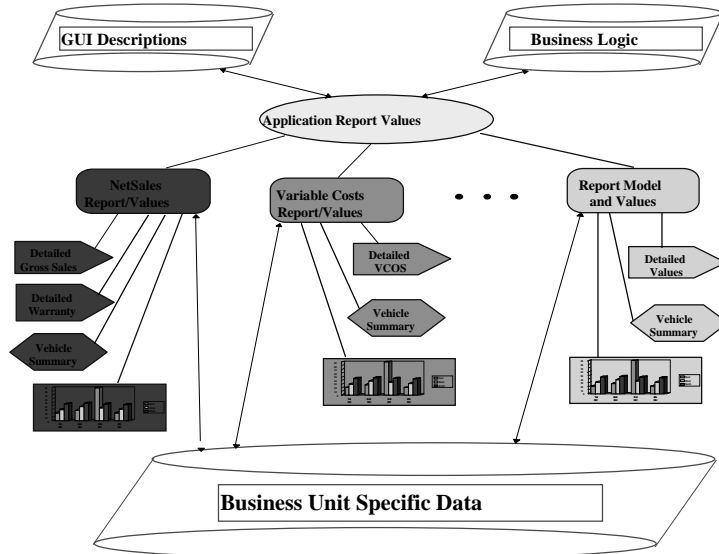
# Solution

- ◆ 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 Joseph W. Yoder & Ralph E. Johnson

56

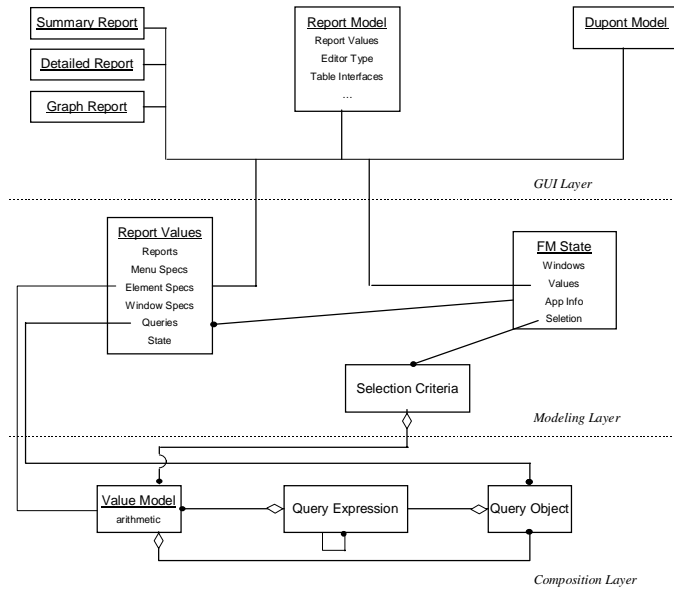
# Report Values



Copyright 1997 by Joseph W. Yoder & Ralph E. Johnson

57

# User View Architecture



Copyright 1997 by Joseph W. Yoder & Ralph E. Johnson

58

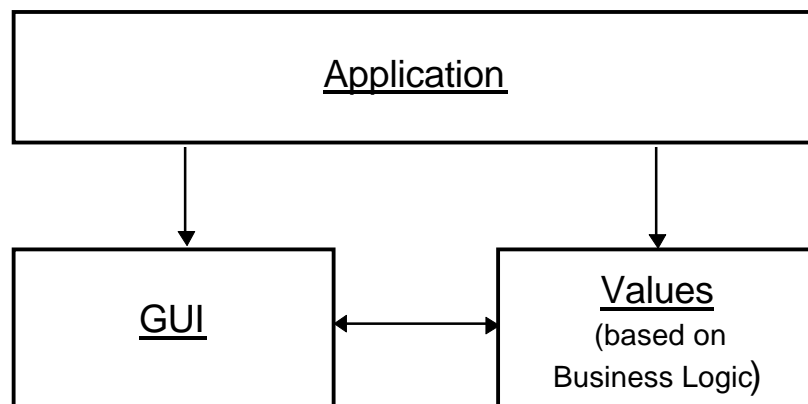
## ReportValue protocol

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

Copyright 1997 by Joseph W. Yoder & Ralph E. Johnson

59

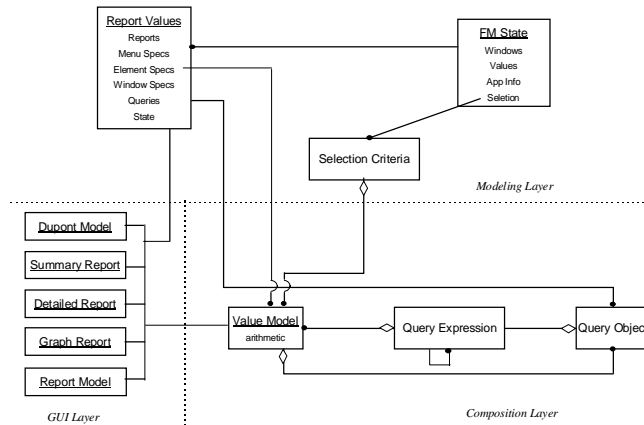
## Layered Architecture



Copyright 1997 by Joseph W. Yoder & Ralph E. Johnson

60

# Black-box Framework



Copyright 1997 by Joseph W. Yoder & Ralph E. Johnson

61

# Visual Builder

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

Copyright 1997 by Joseph W. Yoder & Ralph E. Johnson

62

## Builders

- ◆ Equations in a ReportValue
  - expressions
  - queries
- ◆ GUIs
  - ReportValue (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 Joseph W. Yoder & Ralph E. Johnson

63

## Language Tools

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

Copyright 1997 by Joseph W. Yoder & Ralph E. Johnson

64

## Summary of Architecture

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

## 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

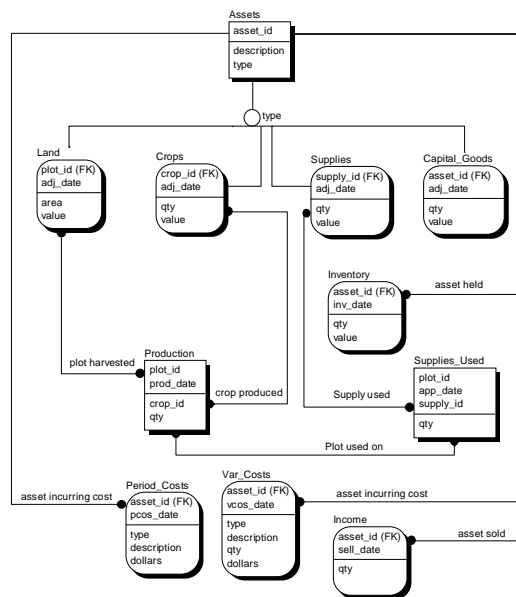
# Data Model

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

Copyright 1997 by Joseph W. Yoder & Ralph E. Johnson

67

# Farm Data Model



Copyright 1997 by Joseph W. Yoder & Ralph E. Johnson

68



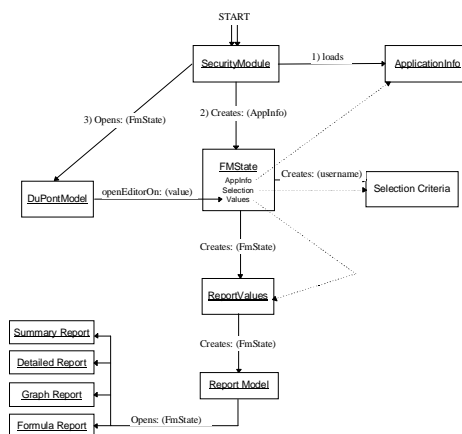
# Security Requirements

- ◆ 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 Joseph W. Yoder & Ralph E. Johnson

71

# FM Creational Diagram



Copyright 1997 by Joseph W. Yoder & Ralph E. Johnson

72

## 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

## 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

## Related Links

- ◆ The following link discusses the details of the framework  
[http://www.uiuc.edu/ph/www/j-yoder/financial\\_framework](http://www.uiuc.edu/ph/www/j-yoder/financial_framework)
- ◆ Good Object-Oriented page with framework references  
<http://st-www.cs.uiuc.edu/users/johnson/>
- ◆ The Evolutionary Patterns Paper - PLoP '96  
<http://st-www.cs.uiuc.edu/users/droberts/evolve.html>
- ◆ The reporting patterns describing query-objects - PLoP '96  
<http://www.uiuc.edu/ph/www/j-yoder/papers/patterns/Reports/>

## Related Links

- ◆ The security patterns used in this framework - PLoP '97  
[http:// www.uiuc.edu/ph/www/j-yoder /papers/patterns/Security/](http://www.uiuc.edu/ph/www/j-yoder/papers/patterns/Security/)
- ◆ Dmitry Zelenko's Masters Thesis describing Query Models  
[http:// www.uiuc.edu/ph/www/j-yoder /papers/thesis/zelenko.ps/](http://www.uiuc.edu/ph/www/j-yoder/papers/thesis/zelenko.ps/)
- ◆ Reflective Facilities and Evolutionary approaches - PLoP '95  
[http:// www.uiuc.edu/ph/www/j-yoder /papers/patterns/Evolution/](http://www.uiuc.edu/ph/www/j-yoder/papers/patterns/Evolution/)
- ◆ Jeff Barcalow's Masters Thesis describing Scenario Planning  
[http:// www.uiuc.edu/ph/www/j-yoder /papers/thesis/barcalow.html](http://www.uiuc.edu/ph/www/j-yoder/papers/thesis/barcalow.html)