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

Goals

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

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

What is a Financial Model?

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

What is a Financial Model?

◆ business logic is equations
  – variable margin = net sales - variable cost
  – net sales = gross sales - warrantee
  – gross sales = sum sales column from sales_and_transfer table
◆ User interface just as important
Warning!
All numbers are fake.

Top Level
Dupont Model

Inventories Drilldown
“Show calculation for Value”

<table>
<thead>
<tr>
<th>Inventories</th>
<th>Budget</th>
<th>Actual</th>
<th>Profit +/-</th>
<th>% Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prime Products</td>
<td>$3,270</td>
<td>$98,650</td>
<td>$77,685</td>
<td>230%</td>
</tr>
<tr>
<td>Production Stores</td>
<td>$63,600</td>
<td>$96,525</td>
<td>$625</td>
<td>2.52%</td>
</tr>
<tr>
<td>INVENTORIES</td>
<td>$83,270</td>
<td>$107,280</td>
<td>$18,710</td>
<td>226%</td>
</tr>
</tbody>
</table>

Copyright 1997 by Ralph E. Johnson
Summary Report

Vehicles by marketing company.

Detailed Transactions

Inspect and edit the individual transactions.
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.htm
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** -
- **ReportModel** - Builds a spreadsheet interface using values and GUI descriptions from *ReportValues*.
- **SummaryReports**
- **DetailedWindows** - Edit and view individual transactions
- **GraphReports**

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:
  ```sql
  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'

ProjectQuery → SelectQuery → TableQuery
#sales_and_transfer

MessageQE
(SUM)

FieldQE
“sales”

MessageQE
(AND)

FieldQE
“family”

ValueQE
'MWL'

ValueQE
'1/1/97'

ValueQE
'1/1/96'

ValueQE
'date'

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
  – Result: twice the classes, some reuse

First Separation

◆ ReportModel
  ◆ SalesModel
  ◆ InventoryModel
  ◆ ...

Uses QueryObjects

◆ ReportValues
  – SalesValues
  – InventoryValues
  – ...

Makes QueryObjects
Three-tiered Client-Server

Database

Business logic

GUI

Model

Controller

Model

Controller

View

Server

Domain Object

Copyright 1997 by Ralph E. Johnson

Structure of Application

View

Application Model

Aspect A

Aspect B

Aspect C

Action A

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

Copyright 1997 by Ralph E. Johnson
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

Window1

- ReportModel1

Window2

- ReportModel2

SelectionBox

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.

```
ProjectQuery  ───► SelectQuery  ───► TableQuery
  ↓                ↓                ↓
MessageQE (SUM)  MessageQE (AND)
  ↓                ↓
FieldQE "sales"  FieldQE "family"
  ↓                ↓
ValueQE '1/1/97' ValueQE 'MWL'
  ↓                ↓
FieldQE "date"    ValueQE '1/1/96'
  ↓                ↓
FieldQE "equal to" FieldQE "less than"
  ↓                ↓
ValueQE '1/1/96'
```

Copyright 1997 by Ralph E. Johnson
Old way - route change through report
  budget value: (query1 values first first).
  actual value: (query2 values first first).
  difference value: budget value - actual value

New way - route change directly to ValueModel
  budget := QueryHolder on: query1
  actual := QueryHolder on: query2
  difference := budget - actual

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

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

Sales Report

<table>
<thead>
<tr>
<th>Sales Report</th>
<th>Budget</th>
<th>Actual</th>
<th>Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$5,000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Thousands of Dollars

Select all sales from North America for the specified date and products which return $5,000,000.
Business logic is equations expressed with ValueModel and QueryObjects

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

\[
\text{variable margin} = \text{net sales} - \text{variable cost}
\]
\[
\text{net sales} = \text{gross sales} - \text{warranty}
\]
\[
\text{gross sales} = \text{sum sales column from sales_and_transfer table}
\]

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.
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.
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

Application

GUI

Values
(based on Business Logic)

Copyright 1997 by Ralph E. Johnson

Black-box Framework

Composition Layer

Modeling Layer

Selection Criteria

Query Expression

Query Object

Listing Values

Menu Specs

Window Specs

Queries

State

Report Values

Graph Report

Report Model

Dupont Model

Summary Report

Detailed Report

Copying layer

GUI Layer

Copyright 1997 by Ralph E. Johnson
Visual Builder

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

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
Language Tools

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

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
Other Features

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

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
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

For more information, see http://cat.ncsa.uiuc.edu/~yoder/financial_framework/