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

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

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

What is a Financial Model?

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

Copyright 1997 by Ralph E. Johnson

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

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

Copyright 1997 by Ralph E. Johnson

Warning!
All numbers
are fake.

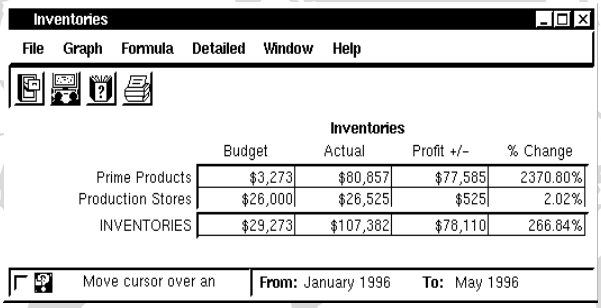
Top Level

Dupont Model

Copyright 1997 by Ralph E. Johnson 7

Inventories Drilldown

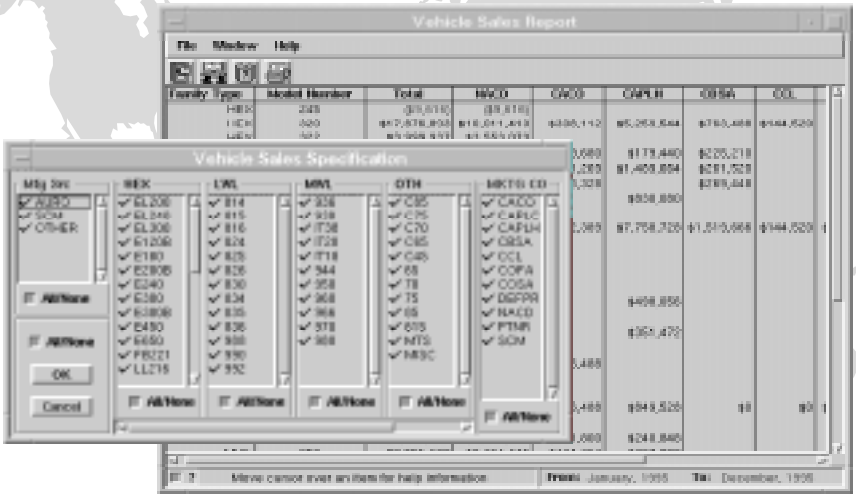
"Show calculation for Value"



Copyright 1997 by Ralph E. Johnson 8

Summary Report

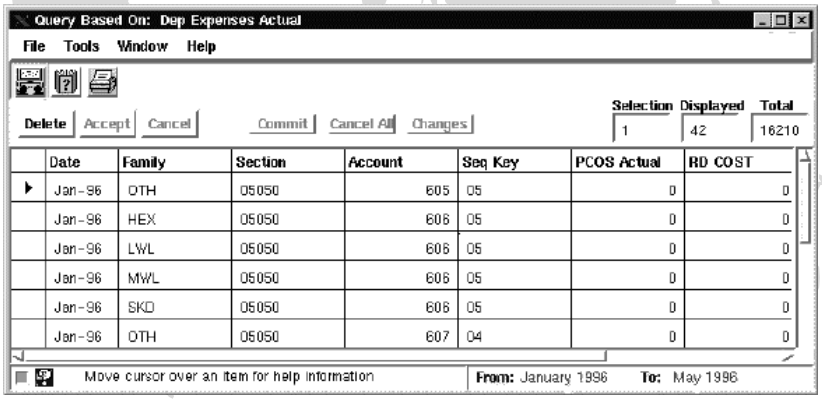
Vehicles by marketing company.



Copyright 1997 by Ralph E. Johnson

Detailed Transactions

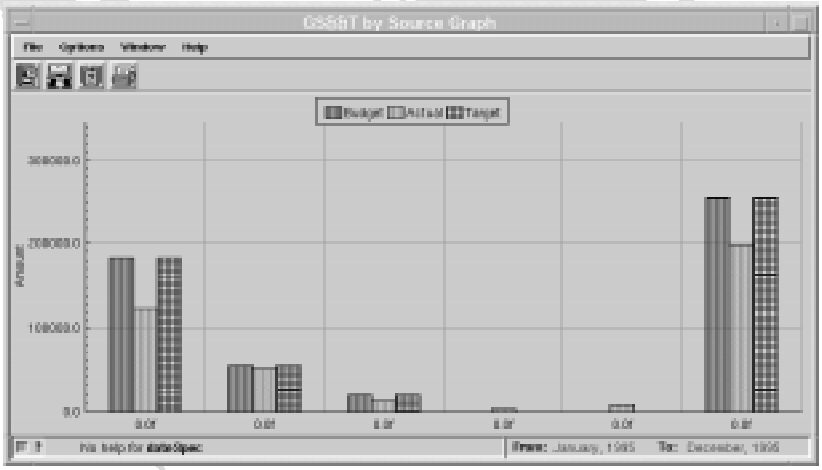
Inspect and edit the individual transactions.



	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

Graphs



Copyright 1997 by Ralph E. Johnson

Summary of Reports

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

Copyright 1997 by Ralph E. Johnson

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

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

Copyright 1997 by Ralph E. Johnson

Three Examples

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

Copyright 1997 by Ralph E. Johnson

White-box Framework

Five kinds of ApplicationsModel, with lots of subclasses

- DupontModel
- ReportModel
- DetailedModel
- SummaryModel
- GraphModel

Copyright 1997 by Ralph E. Johnson

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

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

Component Library

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

Copyright 1997 by Ralph E. Johnson

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

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?

Copyright 1997 by Ralph E. Johnson

Interpreter Pattern

1) make a class hierarchy that represents nodes in abstract syntax tree (SELECT, AND, <, tables, field names)

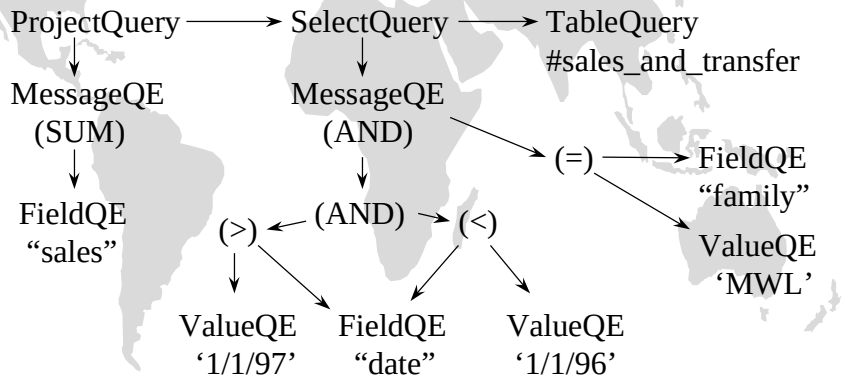
2) define methods to construct and manipulate tree

3) define method to compute value of query (this is the “interpreter”)

Copyright 1997 by Ralph E. Johnson

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

QueryObjects

QueryObject
- TableQuery
- JoinQuery
- WrapperQuery
  - RenamingQuery
  - ExpressionQuery
    - SelectQuery
    - ProjectQuery
    - OrderQuery

QueryExpression
- ValueQE
- MessageQE
- FieldQE
  - RenamedFieldQE

Copyright 1997 by Ralph E. Johnson

QueryObject Protocol

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

Copyright 1997 by Ralph E. Johnson

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
```

Copyright 1997 by Ralph E. Johnson

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

Leading to Black-box

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

Copyright 1997 by Ralph E. Johnson

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

Copyright 1997 by Ralph E. Johnson

First Separation

- ReportModel
  - SalesModel
  - InventoryModel
  - ...

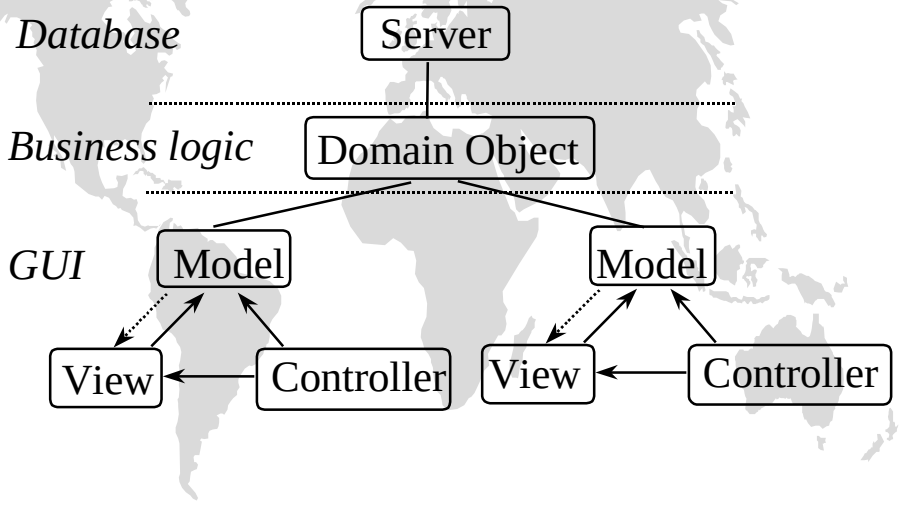
Uses QueryObjects

- ReportValues
  - SalesValues
  - InventoryValues
  - ...

Makes QueryObjects

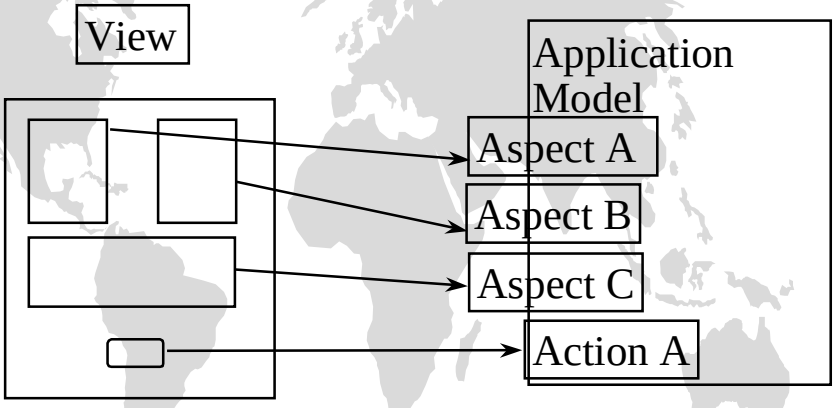
Copyright 1997 by Ralph E. Johnson

Three-tiered Client-Server



Copyright 1997 by Ralph E. Johnson

Structure of Application



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

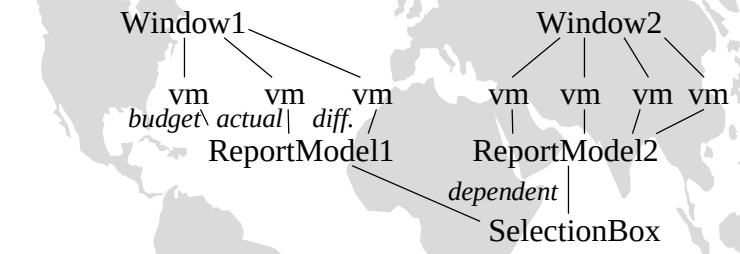
Copyright 1997 by Ralph E. Johnson

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:

Copyright 1997 by Ralph E. Johnson

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.

Copyright 1997 by Ralph E. Johnson

Alternative Solutions

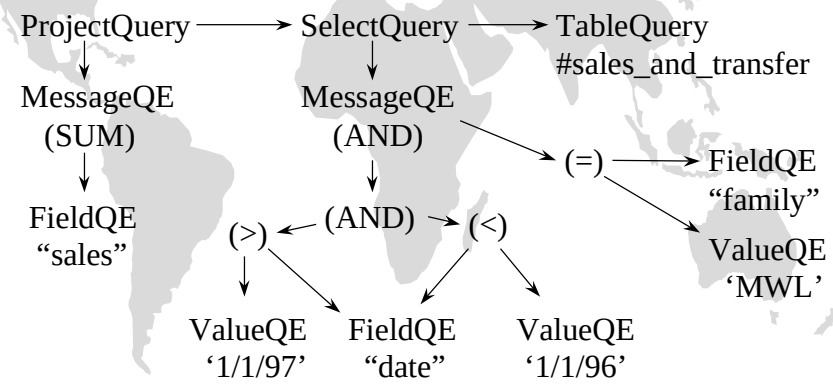
- ReportModel depends on SelectionBox.
  - updates for too many changes
- ReportModel depends on ValueModels from SelectionBox used in QueryObject
  - hard to manage dependencies
- ReportModel depends on QueryObject, QueryObject depends on ValueModels from SelectionBox

Copyright 1997 by Ralph E. Johnson

Observer and QueryObjects

Let ValueQE refer to a ValueModel.

Let each QueryObject observe its components.



Copyright 1997 by Ralph E. Johnson

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 Ralph E. Johnson

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 Ralph E. Johnson

QueryHolder

```
update
    values := aQuery values
    self changed

value
    ^values first first
```

Copyright 1997 by Ralph E. Johnson

Arithmetic on ValueModels

ValueModel implements arithmetic by creating ValueModels that compute function.

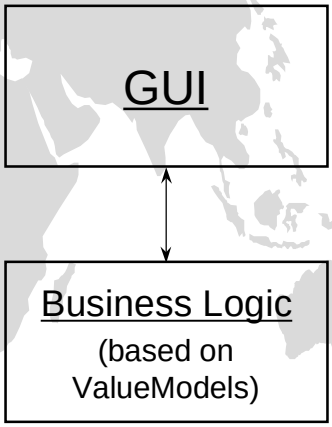
```
+ anObject
    ^BlockValue
        on: [:a :b | a value + b value]
        with: (Array with: self with: anObject)
```

Copyright 1997 by Ralph E. Johnson

Result of Refactoring

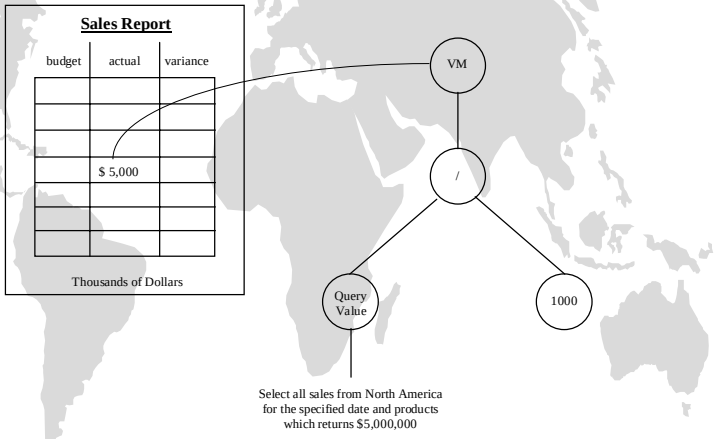
Reuse GUIs, change ValueModels.

Hard part is creating ValueModels and connecting them to GUI.



Copyright 1997 by Ralph E. Johnson

Typical Values in a Report



Copyright 1997 by Ralph E. Johnson

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 Ralph E. Johnson

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

Copyright 1997 by Ralph E. Johnson

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

Copyright 1997 by Ralph E. Johnson

ReportValues

Many tables.

Many columns

Each table has a sequence of valueModels

Total at end.

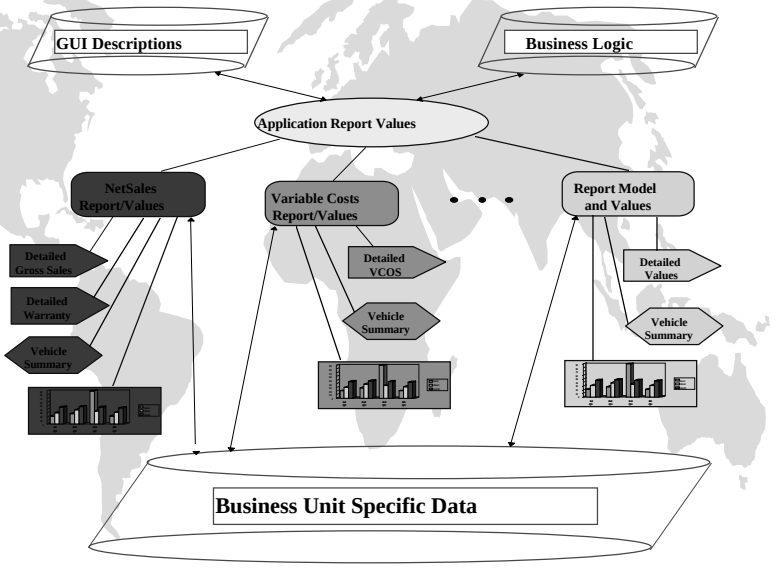
Copyright 1997 by Ralph E. Johnson

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.

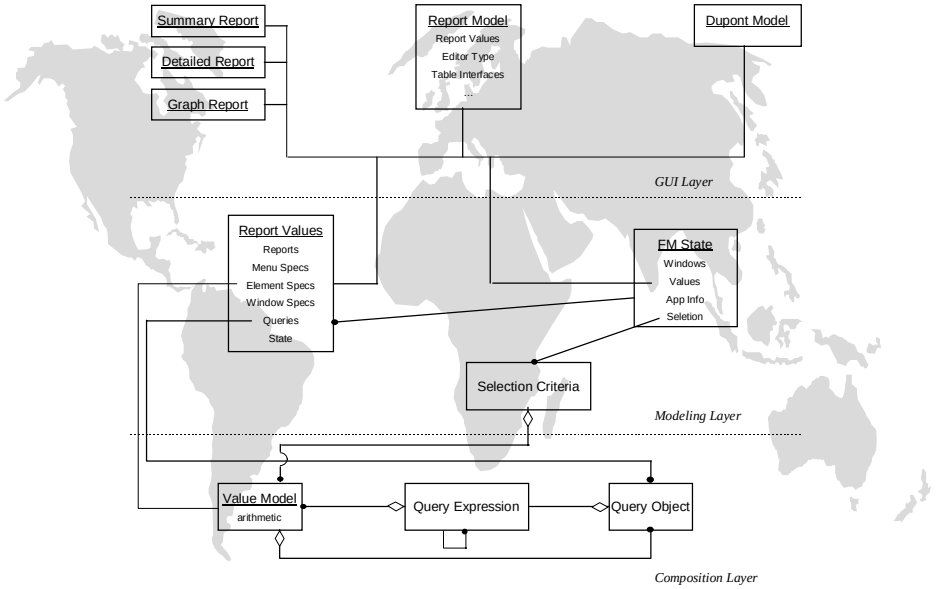
Copyright 1997 by Ralph E. Johnson

Report Values



Copyright 1997 by Ralph E. Johnson

User View Architecture



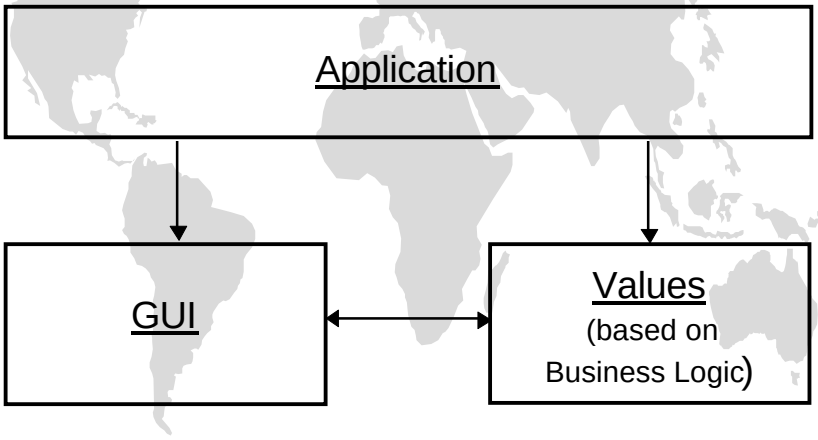
Copyright 1997 by Ralph E. Johnson

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

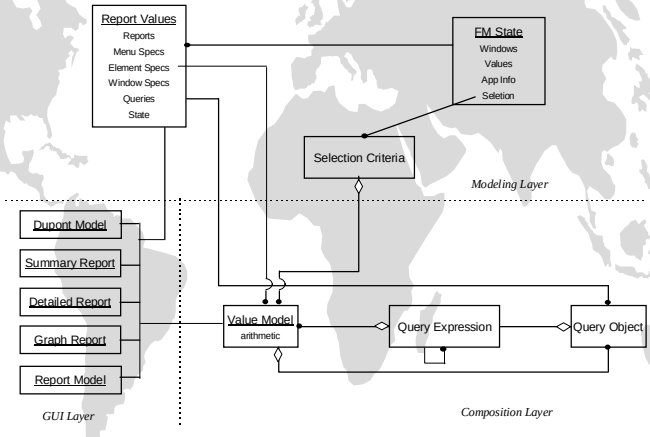
Copyright 1997 by Ralph E. Johnson

Layered Architecture



Copyright 1997 by Ralph E. Johnson

Black-box Framework



Copyright 1997 by Ralph E. Johnson

Visual Builder

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

Copyright 1997 by Ralph E. Johnson

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 Ralph E. Johnson

Language Tools

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

Copyright 1997 by Ralph E. Johnson

Summary of Architecture

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

Copyright 1997 by Ralph E. Johnson

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

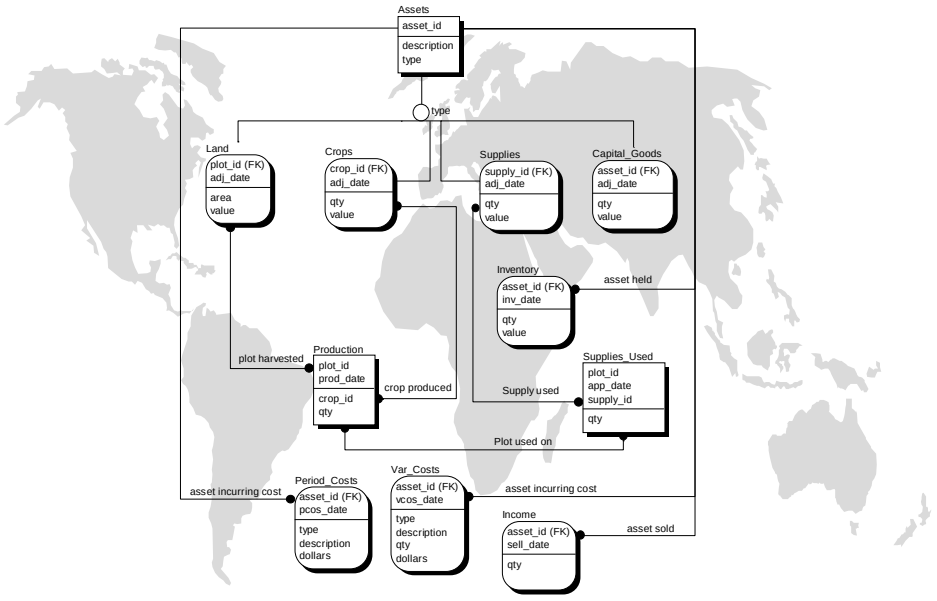
Copyright 1997 by Ralph E. Johnson

Data Model

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

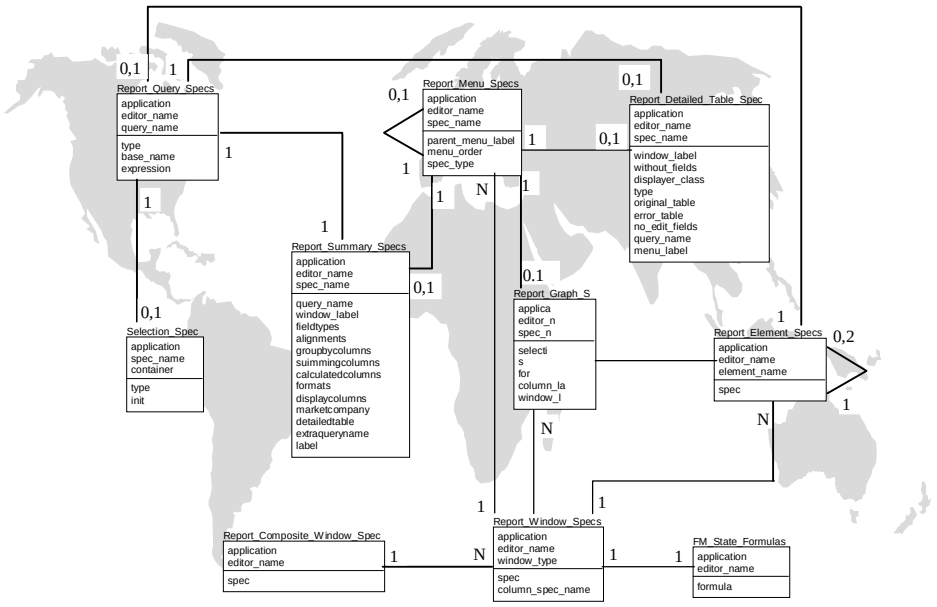
Copyright 1997 by Ralph E. Johnson

Farm Data Model



Copyright 1997 by Ralph E. Johnson

Specification Data Model



Copyright 1997 by Ralph E. Johnson

Other Features

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

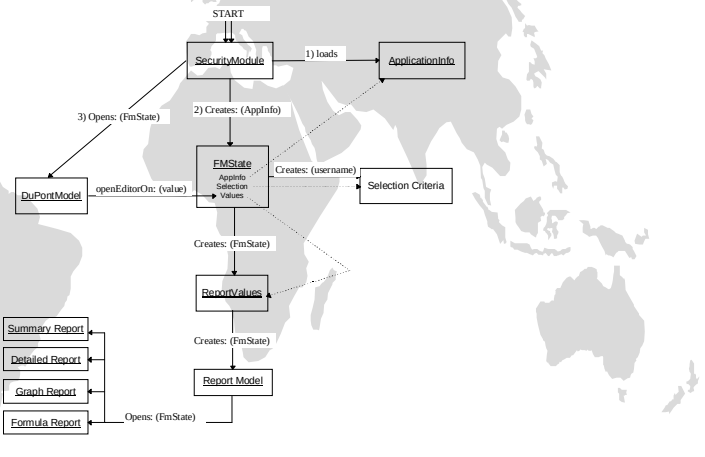
Copyright 1997 by Ralph E. Johnson

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

FM Creational Diagram



Copyright 1997 by Ralph E. Johnson

How to Develop a New Financial Application

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

Copyright 1997 by Ralph E. Johnson

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

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