Patterns in Enterprise Software

Martin Fowler

ThoughtWorks
martinfowler.com

ThoughtWorks
The art of heavy lifting.

Patterns

Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice

Christopher Alexander

» **Chunk of advice**
» **Rooted in practice**
» **Common Knowledge**
  – Helps new people learn
  – Helps old people teach

ThoughtWorks
The art of heavy lifting.

Enterprise Software Patterns

- » **Fowler – *Patterns of Enterprise Application Architecture* – martinfowler.com/eaaCatalog**
- » **Hohpe and Woolf – *Enterprise Integration Patterns* – enterpriseIntegrationPatterns.com**
- » **Hohmann – *Beyond Software Architecture* - lukehohmann.com**
- » **Evans – *Domain Driven Development* – domainLanguage.com**
- » **Alur, Crupi, and Malks – *Core J2EE Patterns***
- » **Microsoft - *Patterns and Practices***
- » **Marinescu – EJB Patterns**

ThoughtWorks The art of heavy lifting.

Enterprise Application

- » **Business Application**
  - – Payroll, health care records, billing, credit scoring, logistics tracking
- » **Large amounts of complex data**
  - – Gigabyte databases with hundreds of tables
- » **Multiple users**
- » **User Interfaces for many roles**
- » **Business logic can be complex and irrational**
- » **Many systems to integrate with**

ThoughtWorks The art of heavy lifting.

3 Points on the Plane

» **B2C Retailer**
  – Catalog with shopping cart
  – Browser UI, many concurrent users, with simple transactions
» **Back-end Leasing**
  – Billing, asset management, accounting
  – Very complex business rules, tens of users, controlled clients
» **Departmental Expense Tracker**
  – Few users, simple rules
  – Must be done very quickly
  – May grow into….

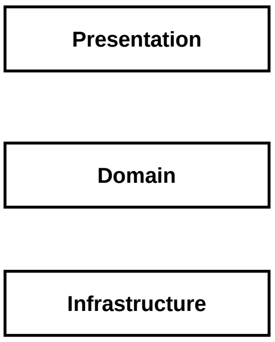
Layered Architecture

✓ **Each layer is a coherent whole**
✓ **Substitute Layers**
✓ **Multiple Higher layers on a lower one**
⌨ **Some things aren't well encapsulated**
⌨ **May harm performance**

Three Primary Layers

- **Presentation**
  - Interacts with the “user” of the application
  - eg: rich client, HTML browser, web service
- **Domain**
  - Business rules, validations, calculations
- **Data Source**
  - Connects to the rest of the enterprise environment
  - Persistence: RDBMs
  - Messaging, TP monitors, legacy apps….

Where to run the layers

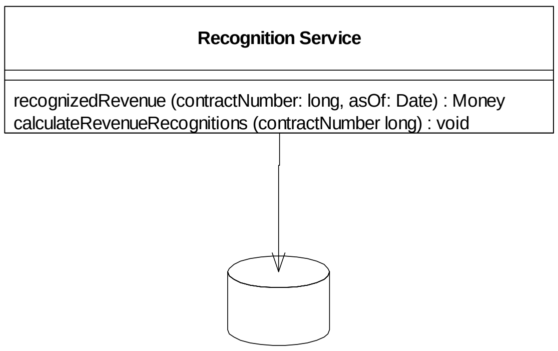


- **Depends on type of system**
- **Minimize Process Boundaries**
  - Often you don’t get the option
- **Running all on server is easiest**
  - ⌨Disconnected operation, responsiveness

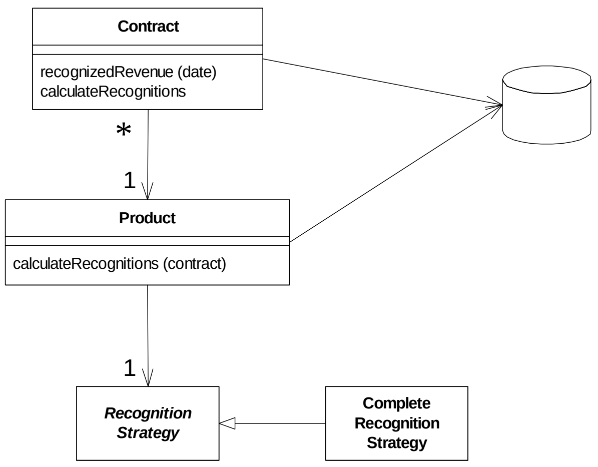
Organizing Domain Logic

- » ***Transaction Script***
- » ***Domain Model***
- » ***(Table Module)***

Transaction Script


Recognition Service
recognizedRevenue (contractNumber: long, asOf: Date) : Money
calculateRevenueRecognitions (contractNumber long) : void

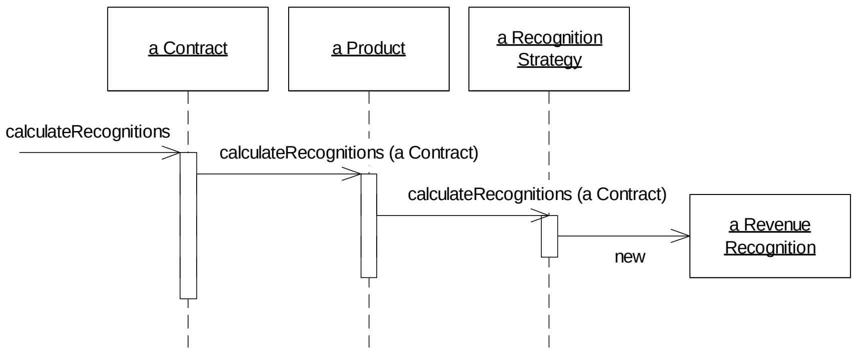
Domain Model



11

ThoughtWorks
The art of heavy lifting.

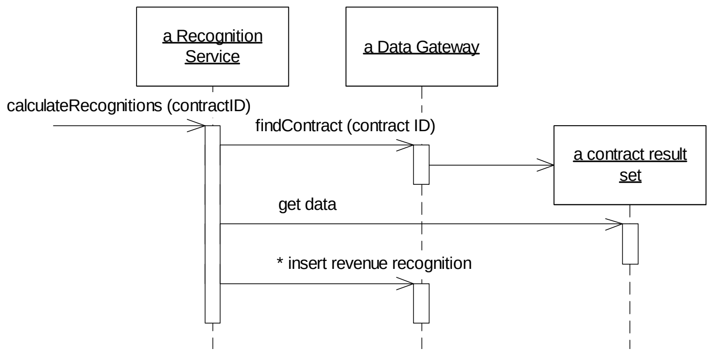
Domain Model: Sequence



12

ThoughtWorks
The art of heavy lifting.

Transaction Script: sequence


a Recognition Service
a Data Gateway
calculateRecognitions (contractID)
findContract (contract ID)
a contract result set
get data
* insert revenue recognition

ThoughtWorks® The art of heavy lifting.℠

Transaction Script: Consequences

- ✓ Simple (Procedural) Programming Model
- ✓ Simple Relationship to database
- ✗ Becomes difficult to work with as domain complexity increases
- ✗ Duplication between scripts

ThoughtWorks® The art of heavy lifting.℠

Domain Model: Consequences

- ✓ Can deal with very complex domain logic
- ⌨ Paradigm Shift
- ⌨ Can have complex mapping to database

Final Thoughts

» Patterns are a mechanism for passing on lessons from different technologies

» The three basic layers are the foundation to an enterprise application architecture

» No pattern is always correct