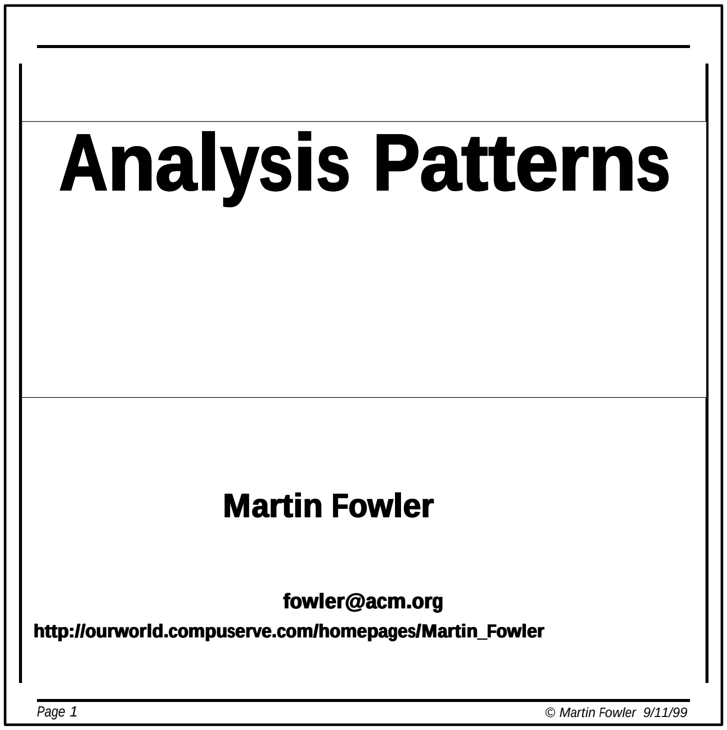
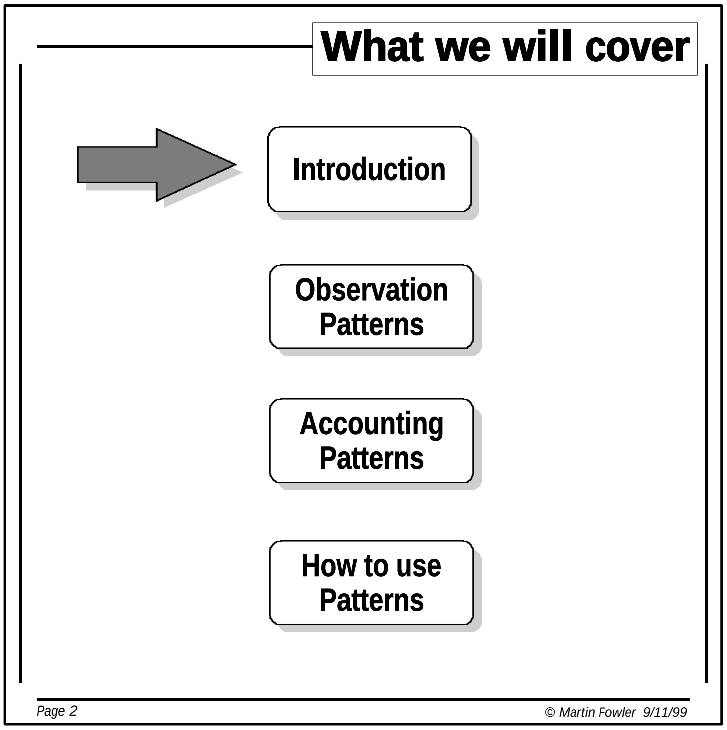


Analysis Patterns

Martin Fowler

fowler@acm.org

http://ourworld.compuserve.com/homepages/Martin_Fowler



What we will cover

- Introduction
- Observation Patterns
- Accounting Patterns
- How to use Patterns

Patterns

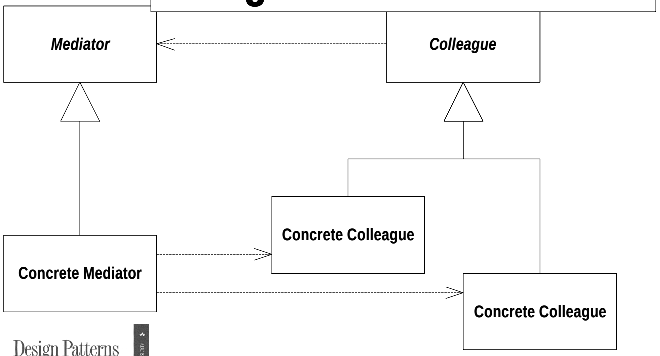
> *...projects fail despite the latest technology for lack of ordinary solutions.*
>
> ***Ralph Johnson and Ward Cunningham***

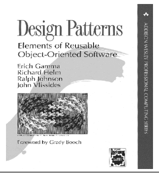
- ❑ **"Gang of Four": Design Patterns book**
- ❑ **Pattern Languages of Programming (PLoP) conferences**

> **A *pattern* is an idea that has been useful in one practical context and will probably be useful in others.**
> **[Fowler]**

http://www.hillside.net/patterns

Design Pattern: *Mediator*





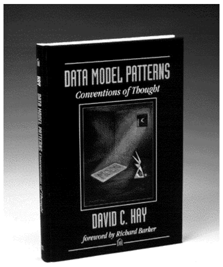
- ❑ **Define an object that encapsulates how a set of objects interact.**

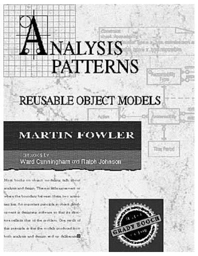
Gamma, E., Helm, R., Johnson, R. and Vlissides, J. *Design Patterns: elements of reusable object-oriented software*, Addison-Wesley, Reading, MA, 1995.

www.industriallogic.com/papers/learning.html

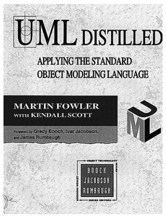
Analysis Patterns

- ❑ **David Hay**
  - ➢ **Relational data models**
  - ➢ **Not connected with patterns community, but still patterns**
  - ➢ **Hay, D. *Data Model Patterns: conventions of thought*, Dorset House, New York, NY, 1996.**
- ❑ **Martin Fowler**
  - ➢ **OO conceptual models**
  - ➢ **Healthcare, Accounting, Financial Trading, Planning**
  - ➢ **Fowler, M. *Analysis Patterns: reusable object models*, Addison-Wesley, Reading MA, 1997.**



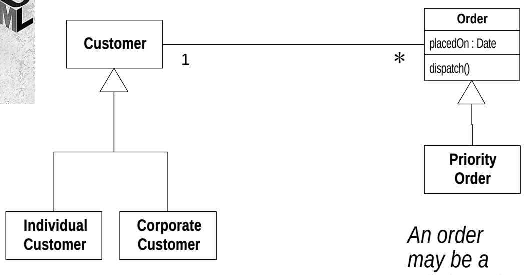


Notation


UML DISTILLED
APPLYING THE STANDARD OBJECT MODELING LANGUAGE
MARTIN FOWLER
WITH KENDALL SCOTT
Grady Booch, Ivar Jacobson, and James Rumbaugh

Each customer may have many orders

Each order has a single customer

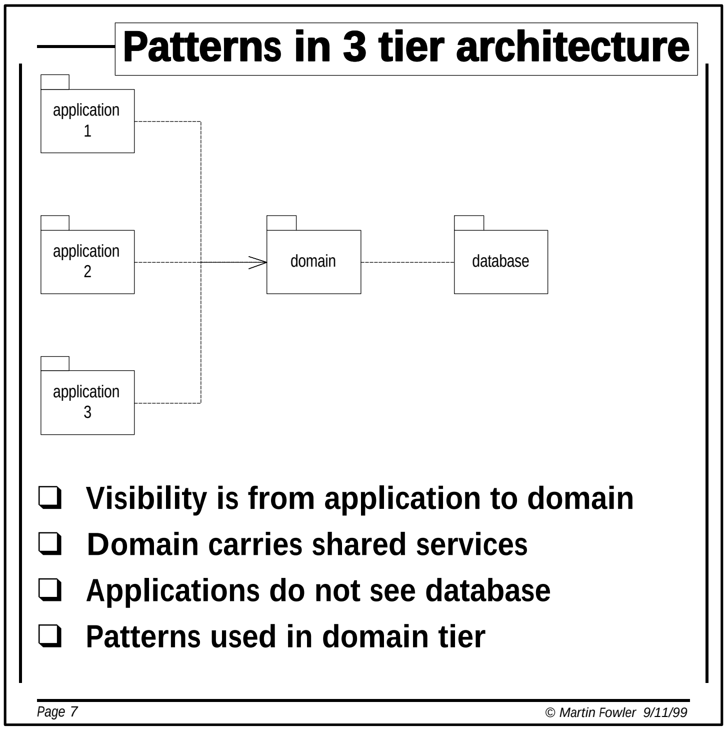


An order may be a priority order

A customer may be a corporate customer or an individual customer

- ❑ **Interface *not* implementation**

Martin Fowler with Kendall Scott, *UML Distilled: Applying the Standard Object Modeling Language*, Addison-Wesley 1997



Patterns in 3 tier architecture

application 1

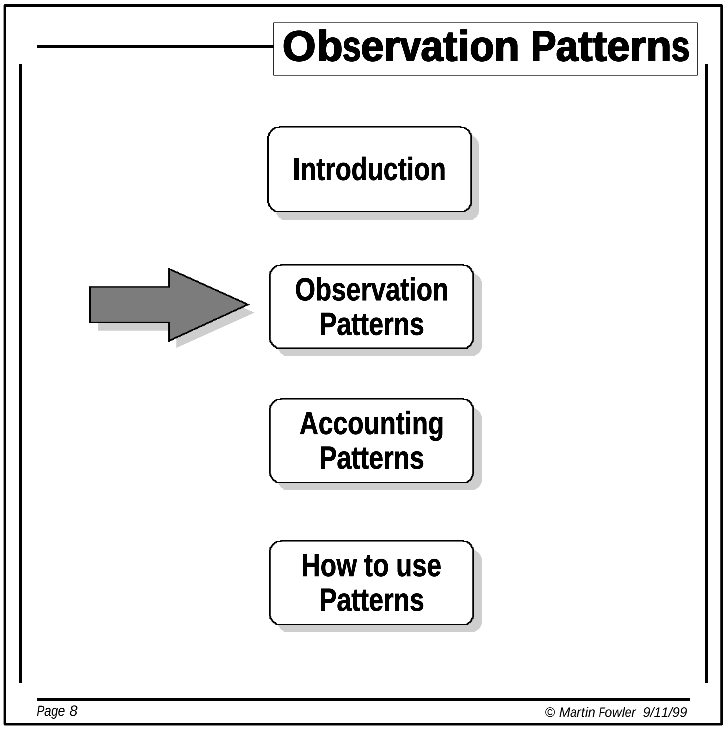
application 2

application 3

domain

database

- ❑ Visibility is from application to domain
- ❑ Domain carries shared services
- ❑ Applications do not see database
- ❑ Patterns used in domain tier



Observation Patterns

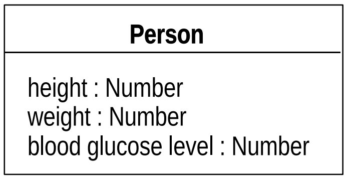
Introduction

Observation Patterns

Accounting Patterns

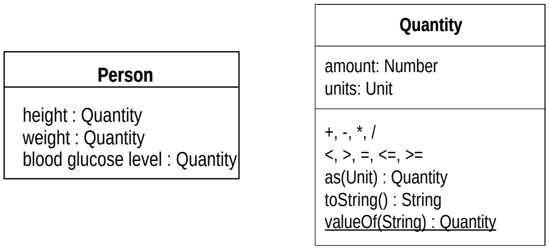
How to use Patterns

Attributes of Person



What are the problems with this model?

Quantity

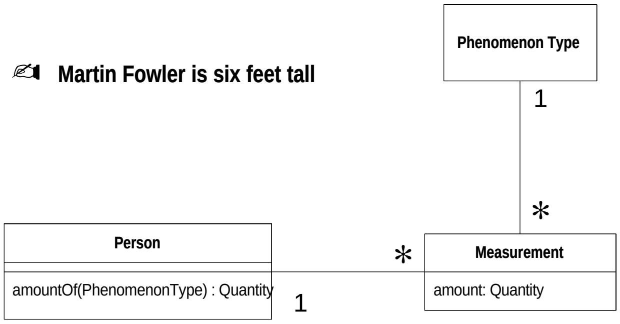


- ❑ **Define arithmetic operations for quantity**
- ❑ **Money is a kind of quantity**
- ❑ **Use *Conversion Ratio* to convert to other units**

What if there are tens or hundreds of attributes?

Measurement

✍ **Martin Fowler is six feet tall**

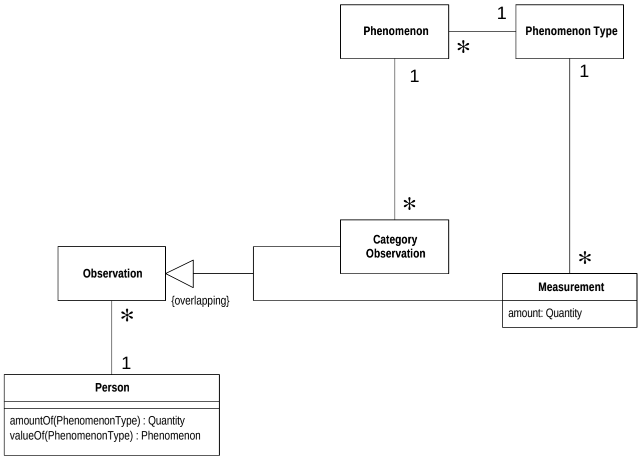


❑ **For each attribute of person, define an instance of phenomenon type**

❑ **To record a value for a person create a measurement**

When should we use this model?

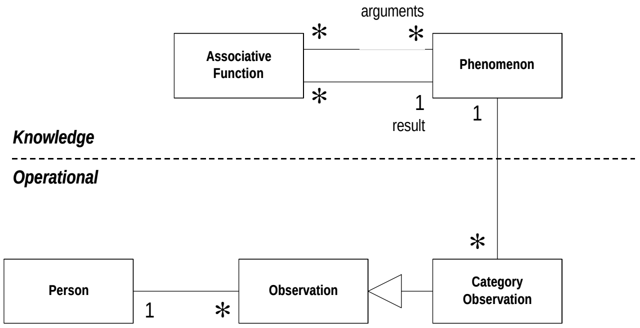
Observation



✍ **Martin Fowler is blood group O**

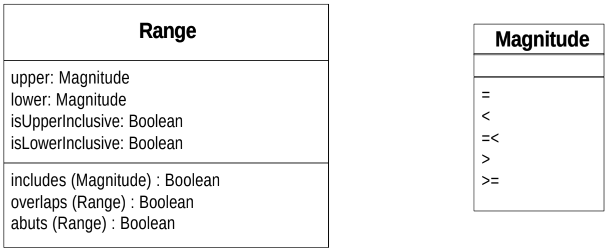
✍ **Martin Fowler is Male**

Knowledge and Operational Levels



- ❑ **Operational**
  - ➢ **regular day to day objects**
- ❑ **Knowledge**
  - ➢ **Meta-Data**
  - ➢ **Objects that capture domain rules**

Range



Range

upper: Magnitude
lower: Magnitude
isUpperInclusive: Boolean
isLowerInclusive: Boolean

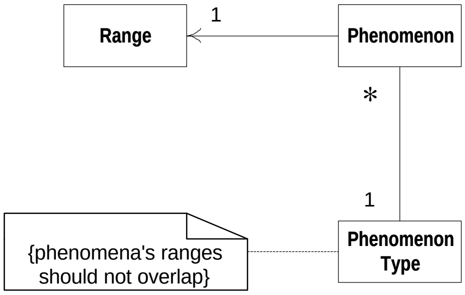
includes (Magnitude) : Boolean
overlaps (Range) : Boolean
abuts (Range) : Boolean

Magnitude

=
<
=<
>
>=

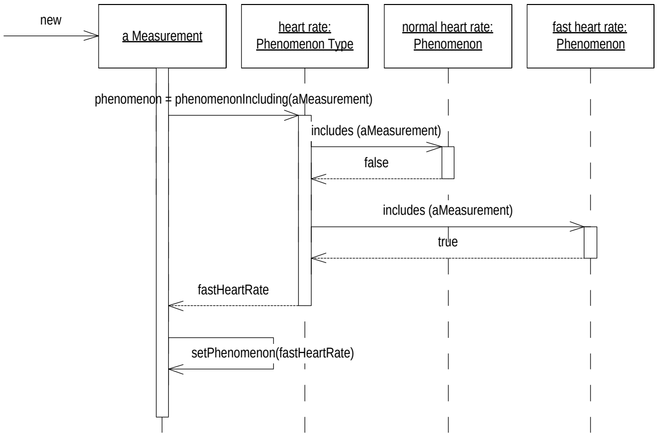
- ✍ **Between 10 miles and 50 miles**
- ✍ **6m < x ≤ 15m**

Phenomenon with Range



✍ **Normal Heart Rate is 60–80 beats per minute**

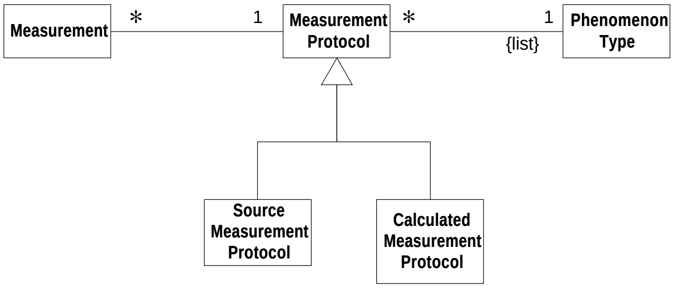
Assigning a phenomenon


new
a Measurement
heart rate: Phenomenon Type
normal heart rate: Phenomenon
fast heart rate: Phenomenon
phenomenon = phenomenonIncluding(aMeasurement)
includes (aMeasurement)
false
includes (aMeasurement)
true
fastHeartRate
setPhenomenon(fastHeartRate)

Corporate Finance Problem

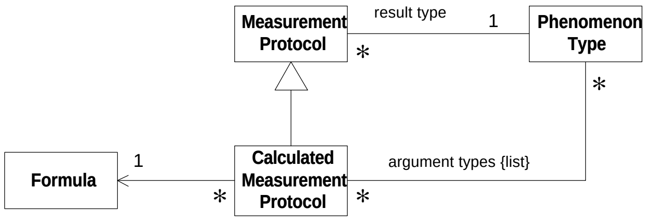
- ❑ Each month we get a large number of indicators (revenue, income, units sold…)
- ❑ Each indicator is given values for good, reasonable, poor, or bad
  - ➢ If revenue is over 105 million that is good, over 95 is reasonable, over 90 is poor and under 90 is bad.
- ❑ Indicators can be calculated based on other indicators
  - ➢ Revenue = income - costs
  - ➢ Income = units sold * average price

Measurement Protocol



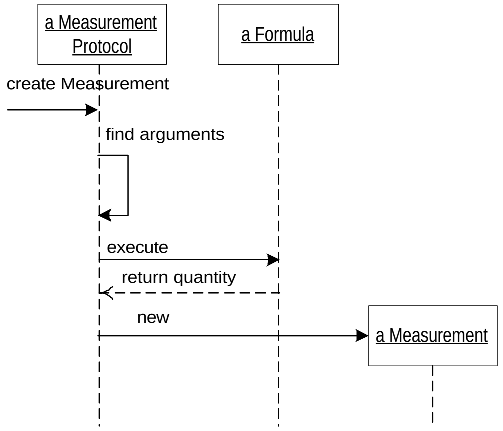
- ❑ Source Protocols read from external systems
- ❑ Calculated Protocols are calculations

Calculated Measurement



- ❑ Calculated measurement protocol has formula and arguments
  - Revenue = Income - Costs
  - Income = Number sold X Average Price
  - Body Mass Index = weight / height^2

Creating a Measurement

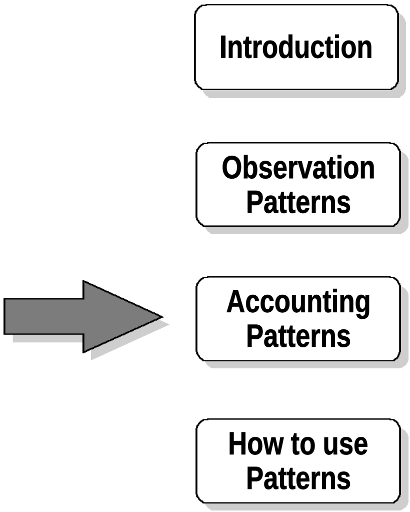


- ❑ Factory method for measurements

Observation Patterns

- ❑ Quantity
- ❑ Measurement
- ❑ Observation
- ❑ Range
- ❑ Phenomenon with Range
- ❑ Measurement Protocol

Accounting Patterns



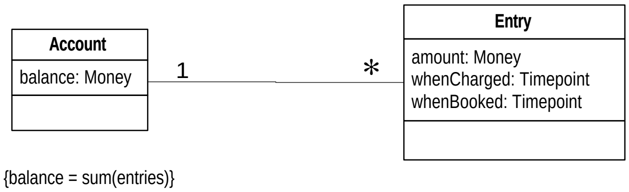
Introduction

Observation Patterns

Accounting Patterns

How to use Patterns

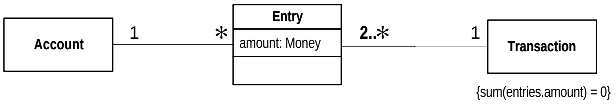
Account



- ❑ **Entries record history of changes to an account**

Where might we use this model?

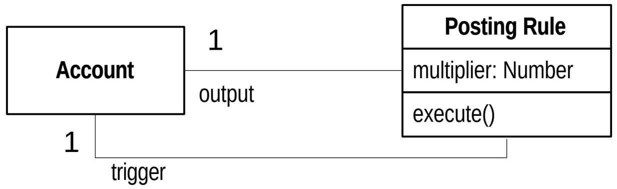
Transaction



- ✍ **I withdraw $100 from my checking account**
- ✍ **The Willenhall Coal Company delivers 30 tons of coal to Bilston Gasworks**

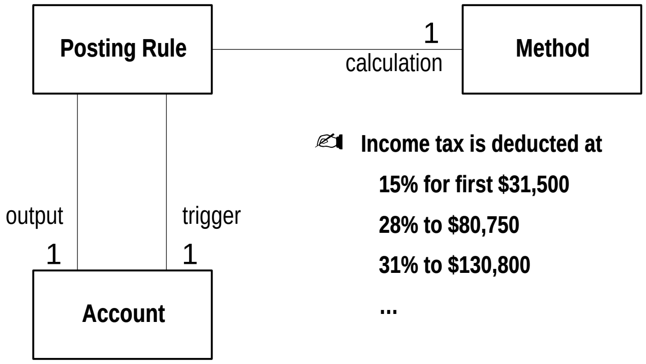
Why are transactions useful?

Posting Rule (simple)



✍ 5% of state tax is deducted from gross income

Individual Instance Method



✍ Income tax is deducted at

15% for first $31,500

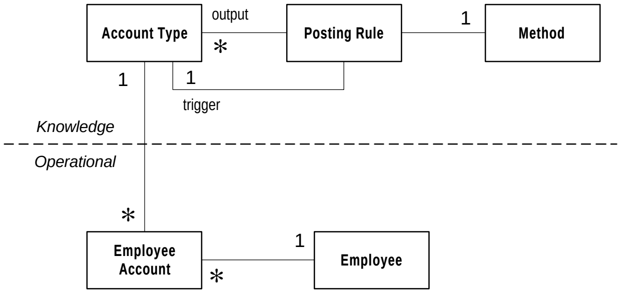
28% to $80,750

31% to $130,800

...

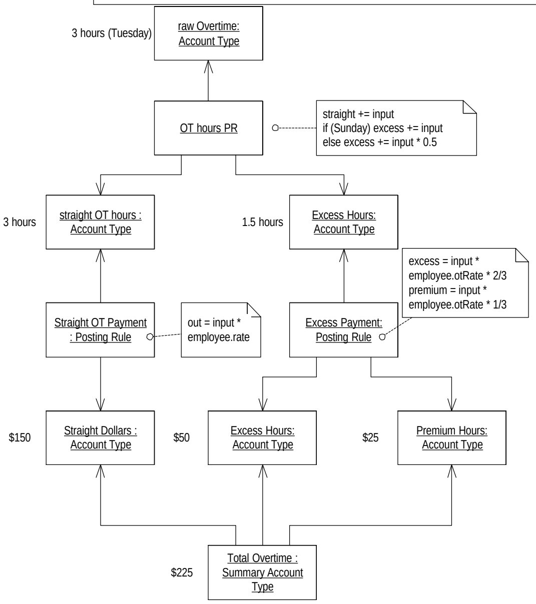
- ❑ Each instance of posting rule has its own method
- ❑ Conceptual statement

Account Types

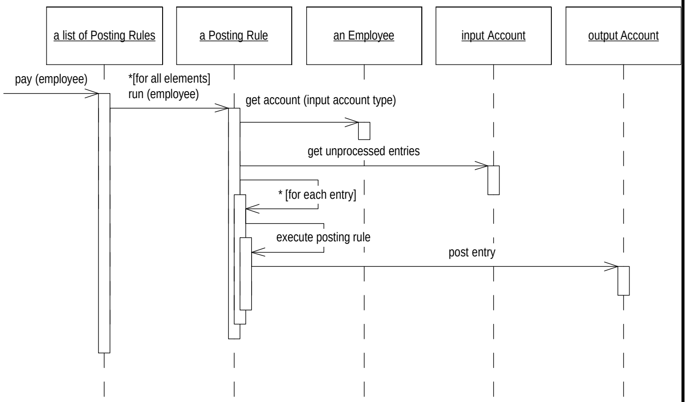


✍ **Apply the graduated tax algorithm to the gross pay of Doug and Dinsdale Piranha to yield their net pay.**

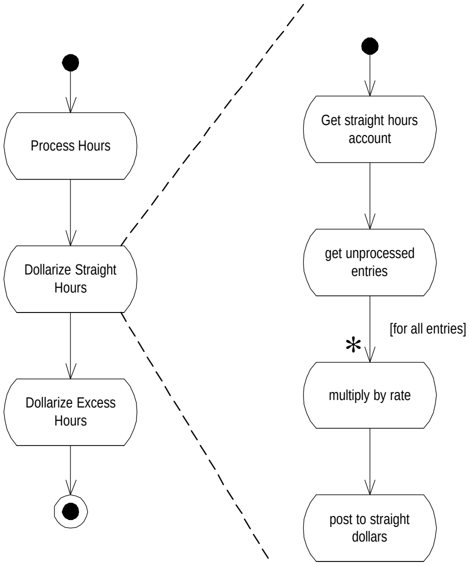
Network of Posting Rules


3 hours (Tuesday)
raw Overtime: Account Type
OT hours PR
straight += input
if (Sunday) excess += input
else excess += input * 0.5
3 hours
straight OT hours : Account Type
1.5 hours
Excess Hours: Account Type
excess = input * employee.otRate * 2/3
premium = input * employee.otRate * 1/3
Straight OT Payment : Posting Rule
out = input * employee.rate
Excess Payment: Posting Rule
$150
Straight Dollars : Account Type
$50
Excess Hours: Account Type
$25
Premium Hours: Account Type
$225
Total Overtime : Summary Account Type

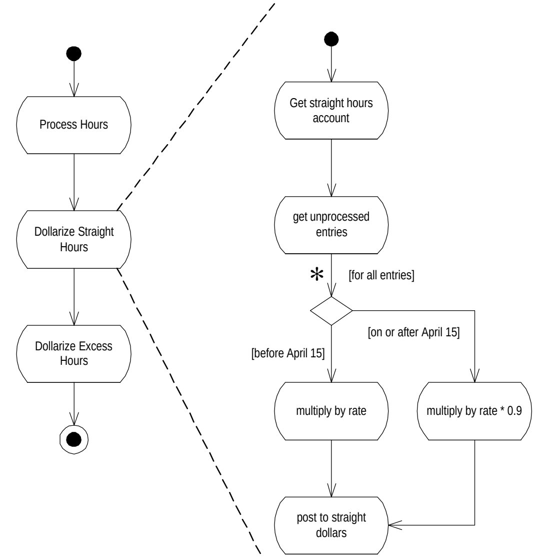
Executing the network



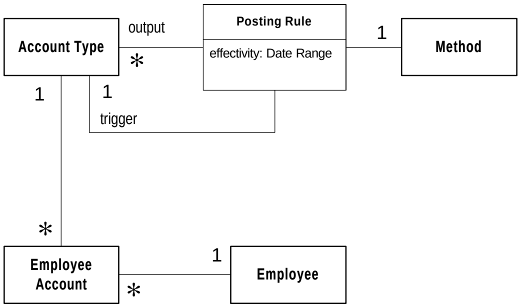
Procedural Approach



Changing the Rules

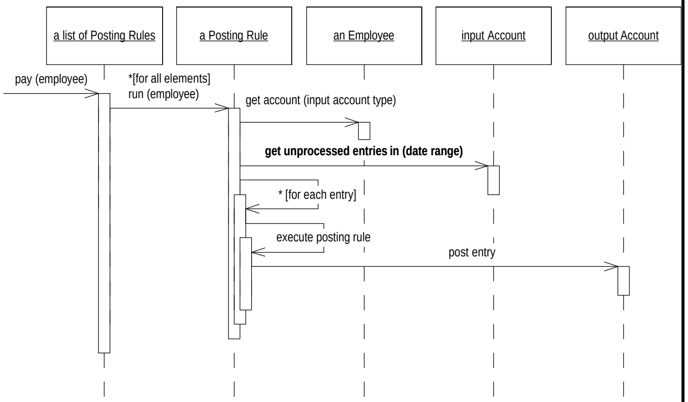


Adding an effectivity time



❑ **Add a date range for each posting rule instance**

New Behavior



a list of Posting Rules

a Posting Rule

an Employee

input Account

output Account

pay (employee)

*[for all elements]
run (employee)

get account (input account type)

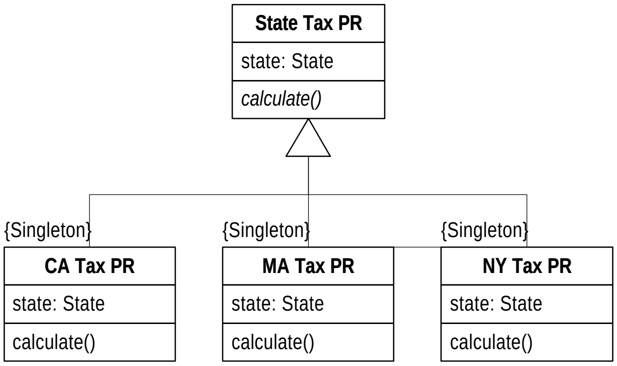
get unprocessed entries in (date range)

* [for each entry]

execute posting rule

post entry

Implementing with singleton



State Tax PR

state: State

calculate()

{Singleton}

CA Tax PR

state: State

calculate()

{Singleton}

MA Tax PR

state: State

calculate()

{Singleton}

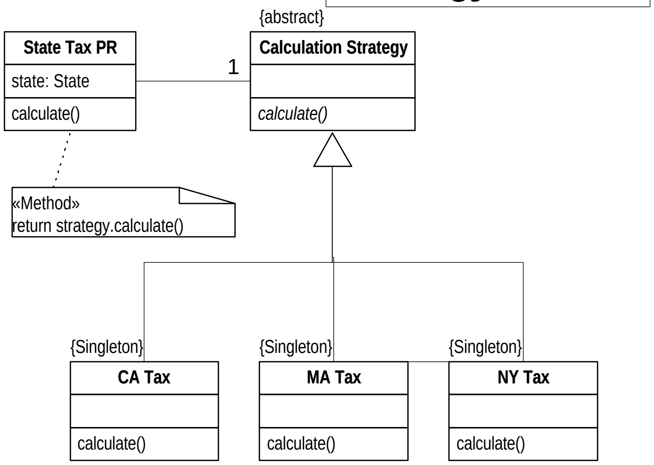
NY Tax PR

state: State

calculate()

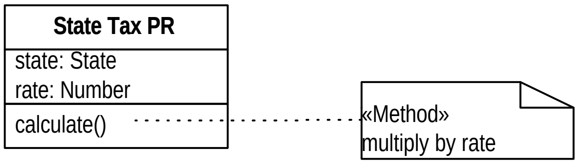
What are the strengths and weaknesses?

Strategy Pattern



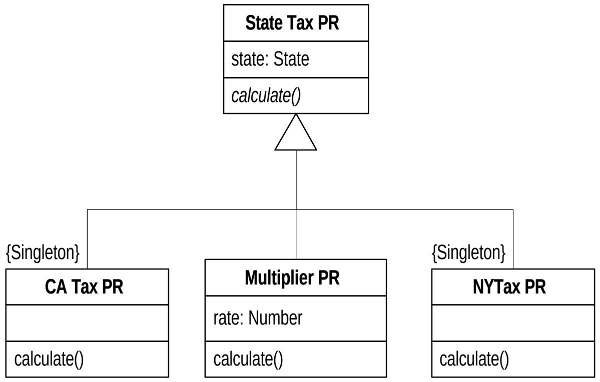
What are the strengths and weaknesses?

Parameterized Method



What are the strengths and weaknesses?

Combining Implementations


State Tax PR

state: State

calculate()

{Singleton}

CA Tax PR

calculate()

Multiplier PR

rate: Number

calculate()

{Singleton}

NYTax PR

calculate()

- ❑ Combination of singleton and parameterized approaches

Posting Rule Execution

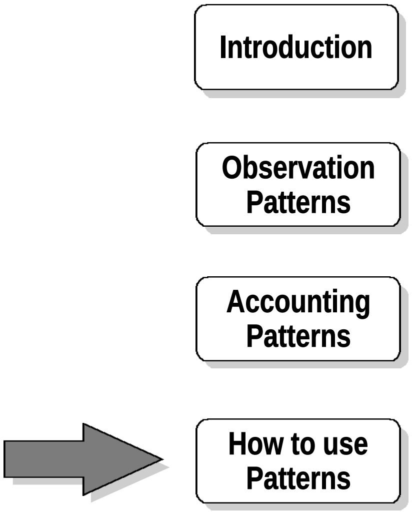
- ❑ Eager firing
  - ➢ Fire when entry added to account
  - ➢ Find all posting rules looking at that account and fire them (etc, etc)
- ❑ Posting rule based
  - ➢ Have a coordinator which tells posting rules when to fire
- ❑ Account based based
  - ➢ Have a coordinator tell which accounts when to fire their outbound posting rules
- ❑ Backward Chained
  - ➢ When you ask an account for its balance
  - ➢ All input accounts must be asked for value (etc, etc)

Which would you choose?

Accounting Patterns

- ❑ Account
- ❑ Transaction
- ❑ Posting Rule
- ❑ Individual Instance Method
- ❑ Posting Rule Execution
- ❑ Posting Rules for Many Accounts
- ❑ Accounting Practice

How to use Patterns

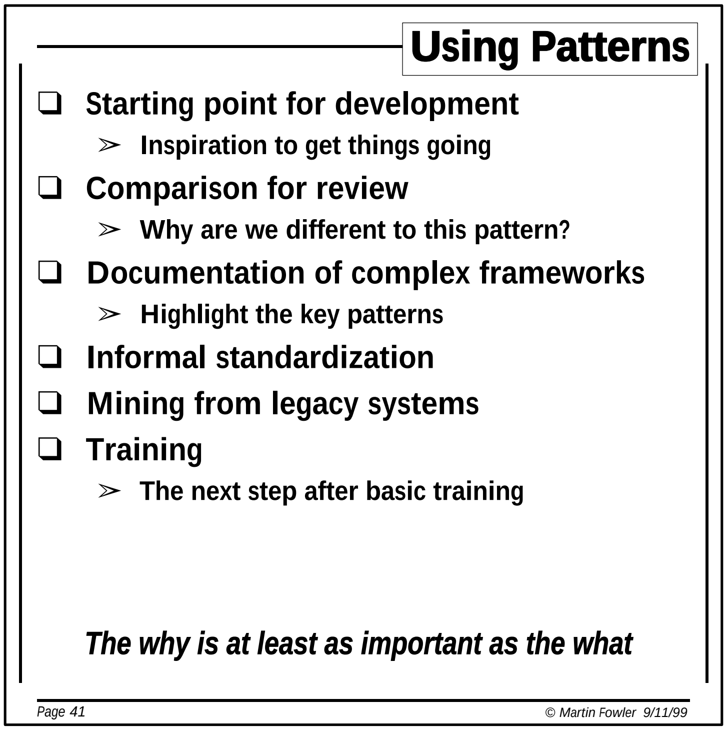


Introduction

Observation Patterns

Accounting Patterns

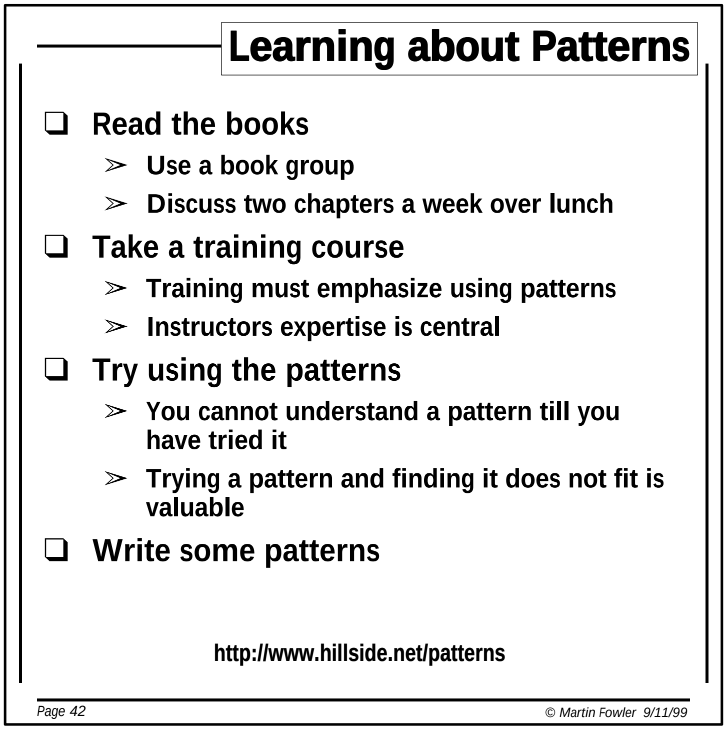
How to use Patterns



Using Patterns

- ❑ Starting point for development
  - ➢ Inspiration to get things going
- ❑ Comparison for review
  - ➢ Why are we different to this pattern?
- ❑ Documentation of complex frameworks
  - ➢ Highlight the key patterns
- ❑ Informal standardization
- ❑ Mining from legacy systems
- ❑ Training
  - ➢ The next step after basic training

The why is at least as important as the what



Learning about Patterns

- ❑ Read the books
  - ➢ Use a book group
  - ➢ Discuss two chapters a week over lunch
- ❑ Take a training course
  - ➢ Training must emphasize using patterns
  - ➢ Instructors expertise is central
- ❑ Try using the patterns
  - ➢ You cannot understand a pattern till you have tried it
  - ➢ Trying a pattern and finding it does not fit is valuable
- ❑ Write some patterns

http://www.hillside.net/patterns

Books

- ❑ Beck, K. ***Smalltalk Best Practice Patterns. Volume 1: Coding***, Prentice Hall, Englewood Cliffs, NJ, 1997.
- ❑ Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P. and Stal., M. ***Pattern-Oriented Software Architecture - A System of Patterns***, John Wiley, 1996.
- ❑ Coad, P., North, D. and Mayfield, M. ***Object Models: strategies, patterns and applications***, Prentice Hall, Englewood Cliffs, 1995.
- ❑ Coplien, J.O. and Schmidt, D.C. ***Pattern Languages of Program Design***, Addison-Wesley, Reading, MA, 1995.
- ❑ Gamma, E., Helm, R., Johnson, R. and Vlissides, J. ***Design Patterns: elements of reusable object-oriented software***, Addison-Wesley, Reading, MA, 1995.
- ❑ Fowler, M. ***Analysis Patterns***, Addison-Wesley, Reading MA, 1997
- ❑ Hay, D. ***Data Model Patterns: conventions of thought***, Dorset House, New York, NY, 1996.
- ❑ Vlissides, J.M., Coplien, J.O. and Kerth, N.L. ***ed.Pattern Languages of Program Design 2***, Addison-Wesley, 1996.

http://www.hillside.net/patterns

Final Thoughts

- ❑ **Patterns are relevant for all aspects of software engineering**
  - ➢ **and for Business Process Reengineering**
- ❑ **Analysis patterns can be used across traditional vertical business domains**
- ❑ **A model is not right or wrong, only more or less useful**
- ❑ **Patterns are a starting point, not a final destination**
- ❑ **Still learning about teaching, using, and discovering patterns**