

# Whither Rules?

# Answer the Following...

- What do we mean when we refer "procedural programming"? What do we mean when we refer to declarative or "rules" programming?
- What are the problem-space characteristics that would lead a developer or architect to choice a procedural vs. declarative approach?
- How does the implementation and future modification of a procedural programs differ from declarative programs?

# Answer the Following...

- How does a business rule approach differ from an object-oriented approach? How can they complement each other?
- What is a "Domain Specific Language" and how does it relate to the choice of procedural vs. declarative programming?
- How does the use of rules fit with SOA?

# Procedural Programming

## Algorithm is King

- Describes actions and order of those actions to reach a desired goal state. Describes "how".
- Focus on maximizing processing resources through modeling a specific solution space.
- Elaborated in generic programming language with a focus on variables, control structure, and looping constructs.

# Procedural Programming

## Algorithm is King

- Best for number crunching, fixed object relationships.
- Poor at symbolic manipulation or relating large groups of objects to each other in variable ways.
- Top-down design paradigm: Detailed requirements, then code.

# Declarative Programming

## Flexible Execution Path

- Describes rules of the process; the 'truths' or the constraints of operations. Not the steps of the process. Describes "what".
- Focus on accurate modeling of the problem space, not the solution space.
- Elaboration using simple truth statements suited to a Domain Specific Language.

# Declarative Programming

## Flexible Execution Path

- Best where execution path or object relationships are highly variable.
- Not specifically optimized for execution performance.
- Bottom-up design paradigm: code each truth as it is uncovered in requirements discovery.

# Procedural Programming

## Problem Spaces

- Engineering or Scientific programming – when we know 'why' and 'why' is immutable.
- Optimization of the solution is critical.
- Where budgets, risk tolerance, and timelines allow a waterfall development approach.
- Future maintenance and review will be conducted by skilled I.T. workers

# Declarative Programming

## Problem Spaces

- Business process modeling – when the underlying motivations are likely to change.
- Solution flexibility is more important than optimization of a particular solution.
- Where budgets, risk tolerance, and timelines demand an agile development approach.
- Future maintenance and review may be conducted by business subject matter experts.

# Object Oriented Programming

- Uses "Objects" and their interactions to implement behavior.
- Supports encapsulation, polymorphism, inheritance, and message-passing as its primary features.
- Business behavior coded as a modular set of message implementations included in the affected object(s).
- Messaging easily supports Event-Driven processing.

# Object Oriented Programming

- Implementing business process within an object creates dependency for a given process with a given version of the object.
- Business processes that involves many objects require abstraction to reduce coupling in order to increase reusability.
- Business process implementation within the objects tends to be procedural in nature.

# Rules and OOP

- Implement business objects to retain state alone.
- Business processes, implemented in rules, act on and update the state of the affected objects.
- Objects are decoupled and highly reusable when they contain no business logic.

# Domain Specific Language

- Modeling business processes and constraints using the language and constructs specific to the subject matter experts of the business processes themselves.
- Define processes in the vocabulary of those responsible for ensuring their effectivity.
- May not be Turing-complete, and therefore may not be able to express any process.

# DSL Example

Elite Status	Miles/Points Required	Segments Required
Executive Platinum	100,000	100
Platinum	50,000	60
Gold	25,000	30

Example from the American Airlines customer information web site

# Example in Java

```
package com.ebdx.drg.example;

public class Customer {
    private int miles;
    private int segments;
    public enum Status {Nonqualifing, Gold, Platinum, Executive_Platinum};

    public Status getStatus() {
        Status status = Status.Nonqualifing;
        if (miles >= 100000 || segments >= 100)
            status = Status.Executive_Platinum;
        else if (miles >= 50000 || segments >= 60)
            status = Status.Platinum;
        else if (miles >= 25000 || segments >= 30)
            status = Status.Gold;
        return status;
    }

    // more code ...
}
```

# Parameter-Driven Implementation

```
package com.ebdx.drg.example;

public class Customer {
    private int miles;
    private int segments;
    public enum Status {Nonqualifing, Gold, Platinum, Executive_Platinum};
    private ArrayList<Threshold> thresholdList;

    public Status getStatus() {
        Status status = Status.Nonqualifing;
        for (Threshold t: thresholdList) {
            if ((miles >= t.getMinMiles() && miles <= t.getMaxMiles()) ||
                (segments >= t.getMinSegment() && segments <= t.getMaxSegment())) {
                status = t.getStatus();
            }
        }
        return status;
    }

    // more code ...
}
```

# Database Update Trigger or Stored Procedure

```
CREATE TRIGGER SETSTATUS BEFORE UPDATE ON Customer
FOR EACH ROW
BEGIN
    IF NEW.MILES >= 100000 || NEW.SEGMENTS >= 100 THEN
        SET NEW.STATUS = 'Executive Platinum';
    ELSEIF NEW.MILES >= 50000 || NEW.SEGMENTS >= 60 THEN
        SET NEW.STATUS = 'Platinum';
    ELSEIF NEW.MILES >= 25000 || NEW.SEGMENTS >= 30 THEN
        SET NEW.STATUS = 'Gold';
    ELSE SET NEW.STATUS = 'Nonqualifying';
    END IF;
END;
```

# Example in Business Rules

Name: Executive Platinum

Documentation:

Definition:

*if*  
the miles of **the customer** is at least 100000  
*or* the segments of **the customer** is at least 100  
*then*  
set the status of **the customer** to Executive Platinum ;

Name: Gold

Documentation:

Definition:

*if*  
the miles of **the customer** is at least 25000 and less than 50000  
*or* the segments of **the customer** is at least 30 and less than 60  
*then*  
set the status of **the customer** to Gold ;

Name: Platinum

Documentation:

Definition:

*if*  
the miles of **the customer** is at least 50000 and less than 100000  
*or* the segments of **the customer** is at least 60 and less than 100  
*then*  
set the status of **the customer** to Platinum ;

Name: NonQualifying

Documentation:

Definition:

*if*  
the miles of **the customer** is less than 25000  
*and* the segments of **the customer** is less than 30  
*then*  
set the status of **the customer** to Nonqualifying ;

# Example in Business Rules

Requirement:

Elite Status	Miles/Points Required	Segments Required
Executive Platinum	100,000	100
Platinum	50,000	60
Gold	25,000	30

Implementation:

	Miles/Points Required	Segments Required	Elite Status
0	$\geq 100,000$	100	Executive Platinum
1	[50,000      100,000[	60	Platinum
2	[25,000      50,000[	30	Gold
3	<i>Otherwise</i>		Nonqualifying

Example shown in ILOG JRules. Other rule engine vendors have similar constructs.

# Domain Specific Language

- Domain specific terms are mapped to classes, attributes, and methods of the underlying Java class.
- Domain terms may be changed without changing the underlying Java implementation:
  - Customer → Guest
  - Employee → Cast Member

# Business Rules & OOP

- In a Business Rules implementation, decision processes act on domain objects which do not include mutable business behavior.
- Mutable business behavior is implemented in the rule definitions and managed through the facilities of the Rule Management System.

# Rules in a Service Oriented Architecture

- Rule sets are best implemented as atomic, reusable business actions required by multiple business processes.
- Rule sets are best housed, managed, and served over a network from an enterprise-wide repository.
- Rule sets are best implemented with discrete, but generalized input and output signatures.

# How Do Rules Fit?

So...

- A Rule Set acts like procedural function: data is passed in as parameters, results are passed out as parameters.
- A Rule Set act like a service: they perform a discrete, reusable business action and are easily available via the network from a central location.

# So How Do Rules Fit?

However, a given Rule Set is defined as a group of declarative rules. This provides these advantages over alternative approaches:

- Flexible within a given business action
- Consistently reusable within a business process
- Easily understood, reviewable, and maintainable by business subject matter experts
- Easily expressed in a Domain Specific Language

# Questions?

Lawrence Terrill  
EBDX.COM  
rules@ebdx.com



This presentation is licensed under a  
Creative Commons Attribution-Noncommercial-Share Alike 3.0 United States License.