



**MERCURY™**

# **Business Processes**

## **Connecting the Design-Time to the Run-Time**

IBM Programming Languages & Development Environments Seminar

Nov. 30, 2005

**Joshua Fox** יהושע פוקס

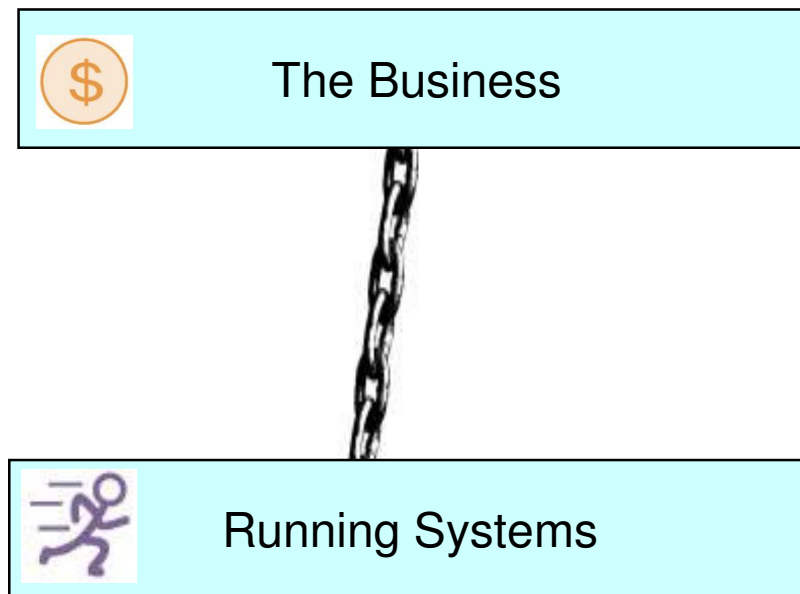
Mercury Interactive

# Agenda

- The goal
- The challenges
- Solutions

# The Goal

- Connect the business process with runtime behavior



# Business Value

We must understand:

- Performance problems
- Bugs
- Functional incompleteness
- Functional errors: The software's functionality is different from business needs

# Case Study #1


- Equity-pricing software
- Showing suspiciously low-priced sales
- Suspicion: Security failure
- Actual cause: Complex business rule for discounts to favored customers, which recently started kicking in far more often
- Result: Change the pricing algorithm (Change in business)

## Case Study #2


- Programmatic Trading software: Multiple bugs or performance bottleneck
- Not possible for developers to fix the bug or optimize the code
- Take this to business level
  - Reanalyze the business domain
  - Simplify the model
  - Simplify Algorithm

# The Layers


## Examples

 The Business


Real Life

 Abstract BP Models

BPMN

 Functional Model/Code

BPEL, Java

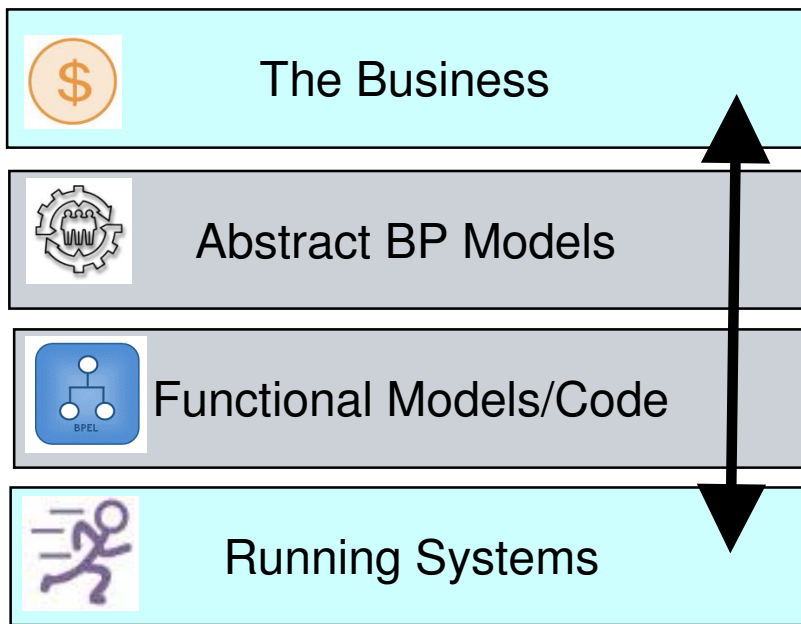
 Running Systems

 Monitoring

Execution

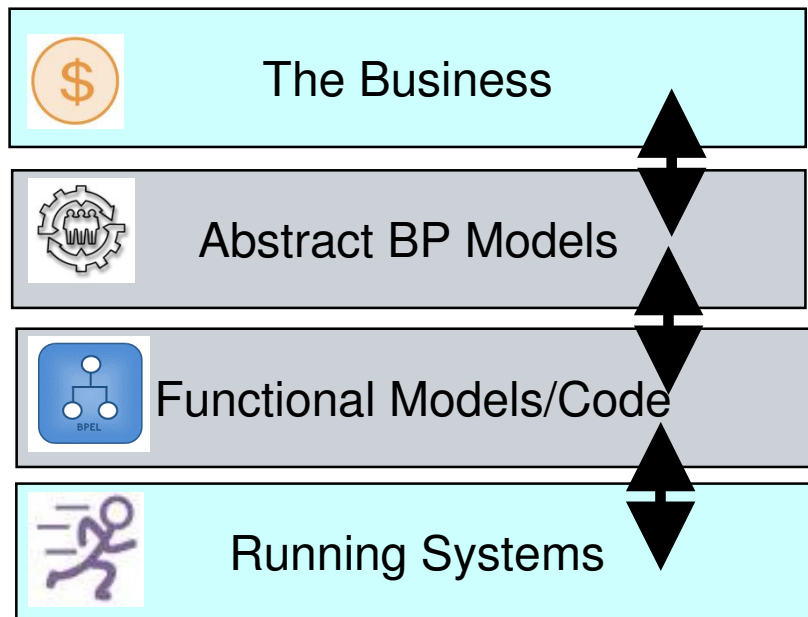
Profilers

# Achieving Coherence



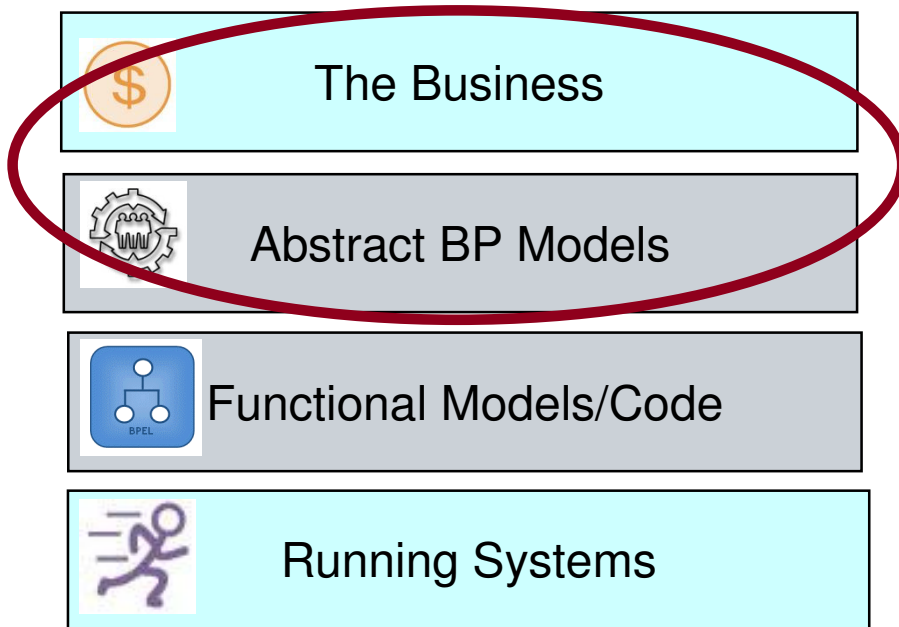
- Connect all the layers
- Layered architecture valuable only when each layer plays its role in relation to the ones above and below
- When each layer connected up and down, we can trace the connection between any two layers
- Ultimately connect the top and bottom layers: The Business and the software runtime

# The Challenges



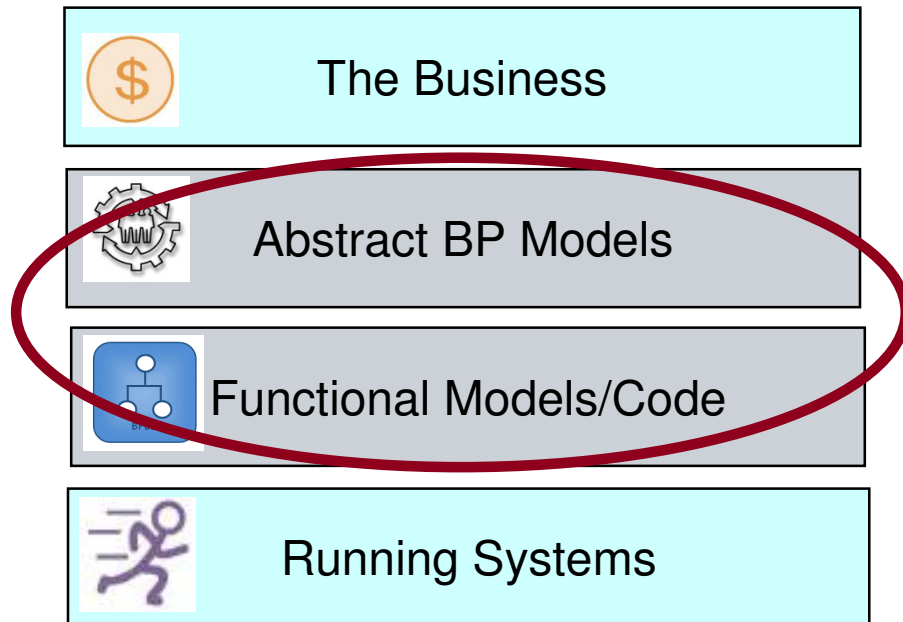
- Each layer needs to be connected to the one below it
- Not all layers can be generated directly
- Layers use different human/software languages
- Language aside, the concepts of each layer are different
- Biggest gap: Design-time vs. Runtime

# Existing Solutions



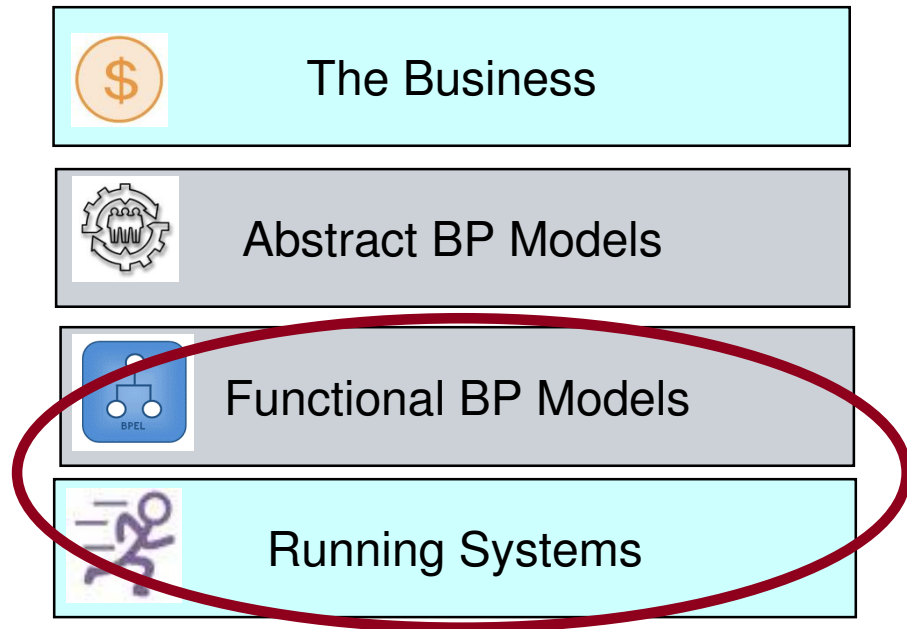
- Business Process Modeling Notation
- Formalizing the business

# Existing Solutions



- Business Process Execution Language
- BPMN → BPEL Mapping specified in BPMN standard
- Linking to execution in Web Services, Java
- See Article by IBM's S. White
- See BP products from Telelogic, iGrafx, IDS Scheer/Arise


# Existing Solutions





- BPEL Engines
- Java Virtual Machines

# Existing Solutions

 The Business

 Abstract BP Models

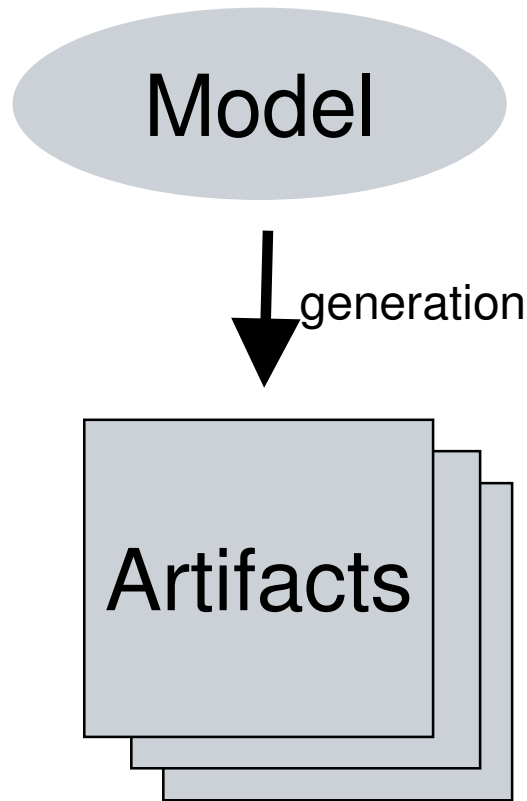
 Functional Models/Code

 Running Systems

 Monitoring

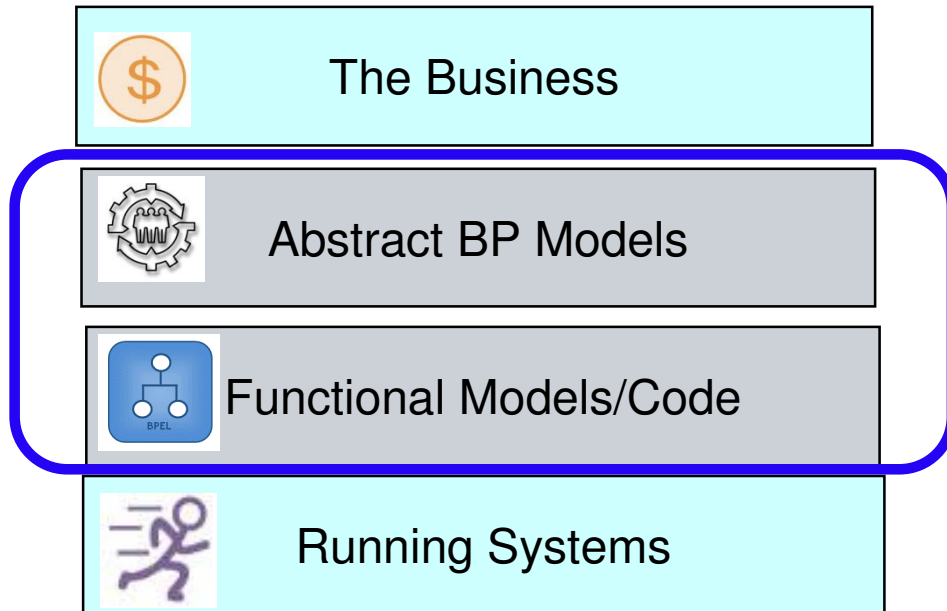
- Logging
- Debugger
- Monitoring Tools
- Appserver Monitor
- Profiler
- Diagnostics

# Model Driven Architecture



- Build your model
- Generate
  - Executable code in any software language
  - Database schemas
  - Messaging formats
- Value
  - Internally coherent system
  - Align code with model

# MDA - Challenges



- Usually a static rather than BP model
- Corresponds only to layers down to Execution Code, and so:
  - creates consistency between codebase/metadata and the business (model),
  - but not down to the runtime behavior

# Annotations: What They Are

- Added in JDK 5.0 (1.5)
- Tags added to class, interface, method, etc.
- The tag-definitions themselves are in interface-like files

```
@BPTask("Shipping/dropship")  
public void send(Product p) {  
}
```

# Annotations: What They Do

- *Declarative* statements in the code:  
Constraints or behavior
- Can be set to compile into the bytecode
- Can be used to instrument code

# Annotations: A Link in the Chain

- Added either
  - during MDA-style code generation
  - or manually
- Connecting the BP model to the runtime monitoring tools
- Generate code that can declare what part of the process the code implements

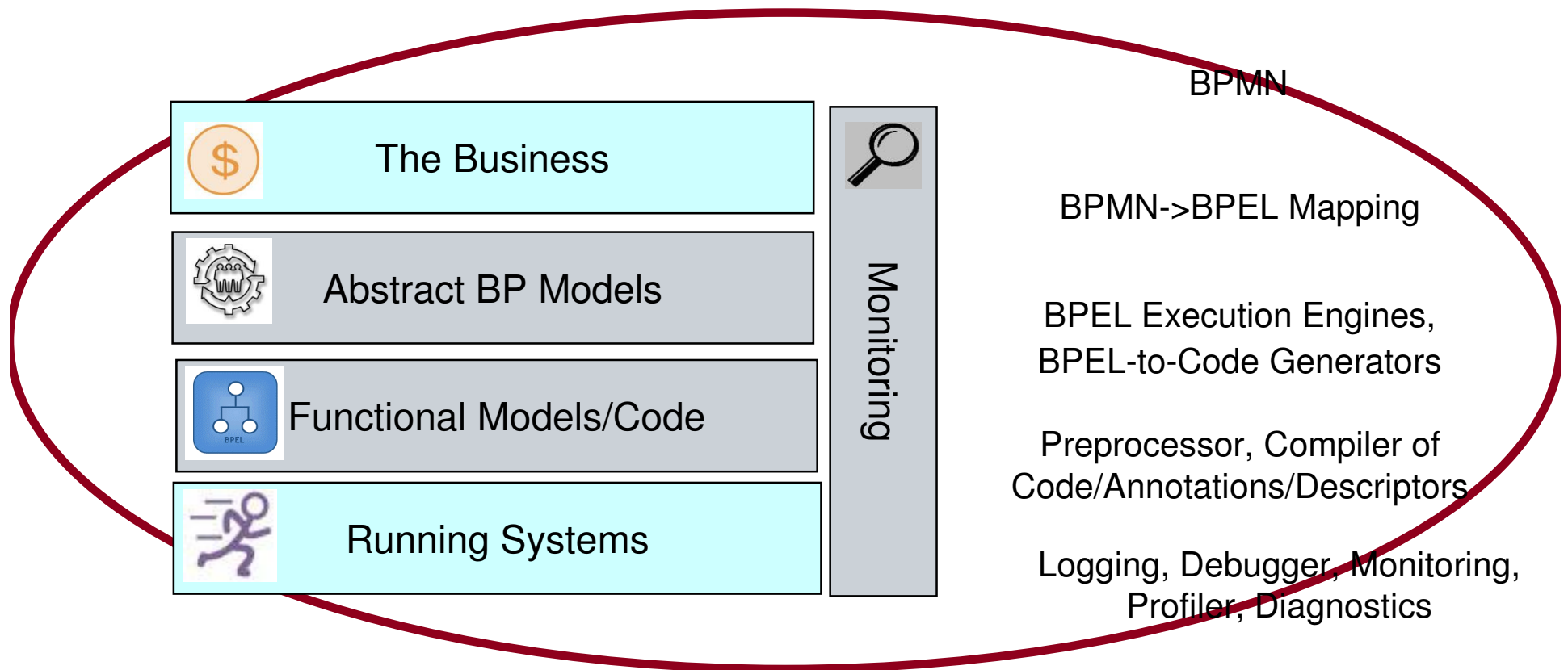
## External Association

- Before Annotations came XML Configuration files
- E.g., EJB Deployment Descriptors
- Like annotations, can be used to connect legacy/non-generated code with model
- Can be used to guide the runtime instrumentation of the class files
- Similarities to Aspect Oriented Programming

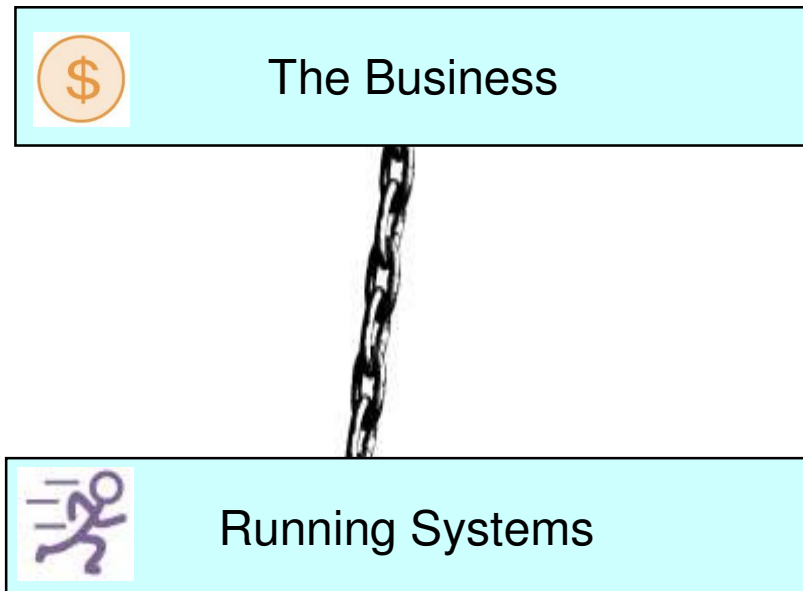
# Applying This to Other Technologies

- .NET has annotations
- Various other languages have the equivalents
- Where no annotations available, options include
  - Preprocessors
  - Generated or Manual external files

# Monitoring: From Runtime to Business



# Conclusions



- *Running Software serves the Business*
- Connect running software to the business
- Help the software achieve business goals

## Contact

- [jfox@mercury.com](mailto:jfox@mercury.com)

**MERCURY™**