

---

# Action Specifications for Unified Modeling Language

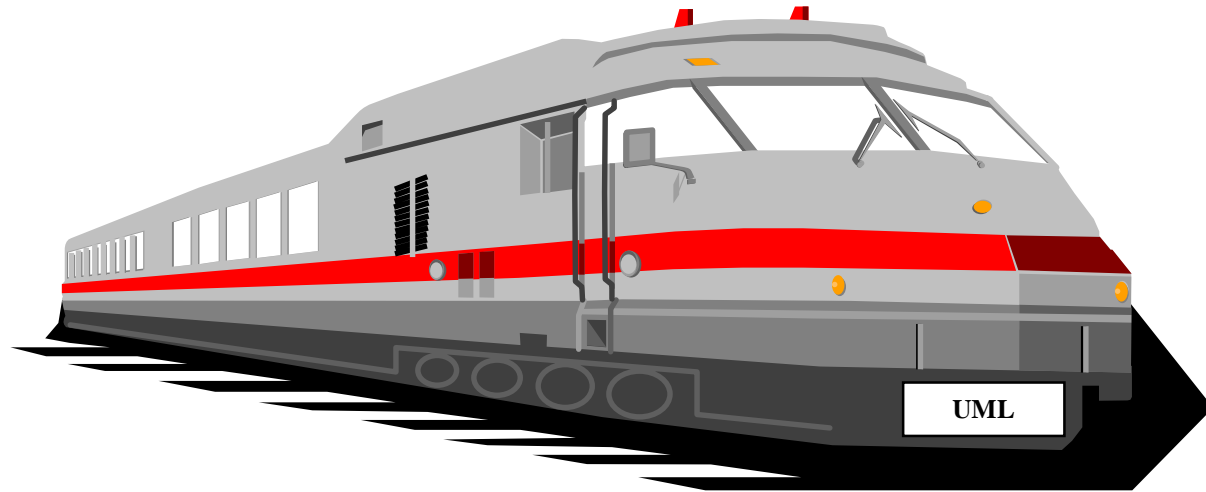
Stephen J. Mellor  
Project Technology, Inc.  
<http://www.projtech.com>  
[steve@projtech.com](mailto:steve@projtech.com)  
+1 (510) 567-0255 x619

Michael M. Lee/Dirk Epperson  
Softwire, Inc.  
<http://www.softwire.com>  
[mike/dirk@softwire.com](mailto:mike/dirk@softwire.com)  
+1 (415) 925-3481

# Congratulations!

---

The UML is now an  
OMG standard!



# Real-World Applications Follow

---

Hmm.. I have my Use Cases,  
Packages, Objects,  
Associations,  
State Models, Actions and  
more. ... Good,

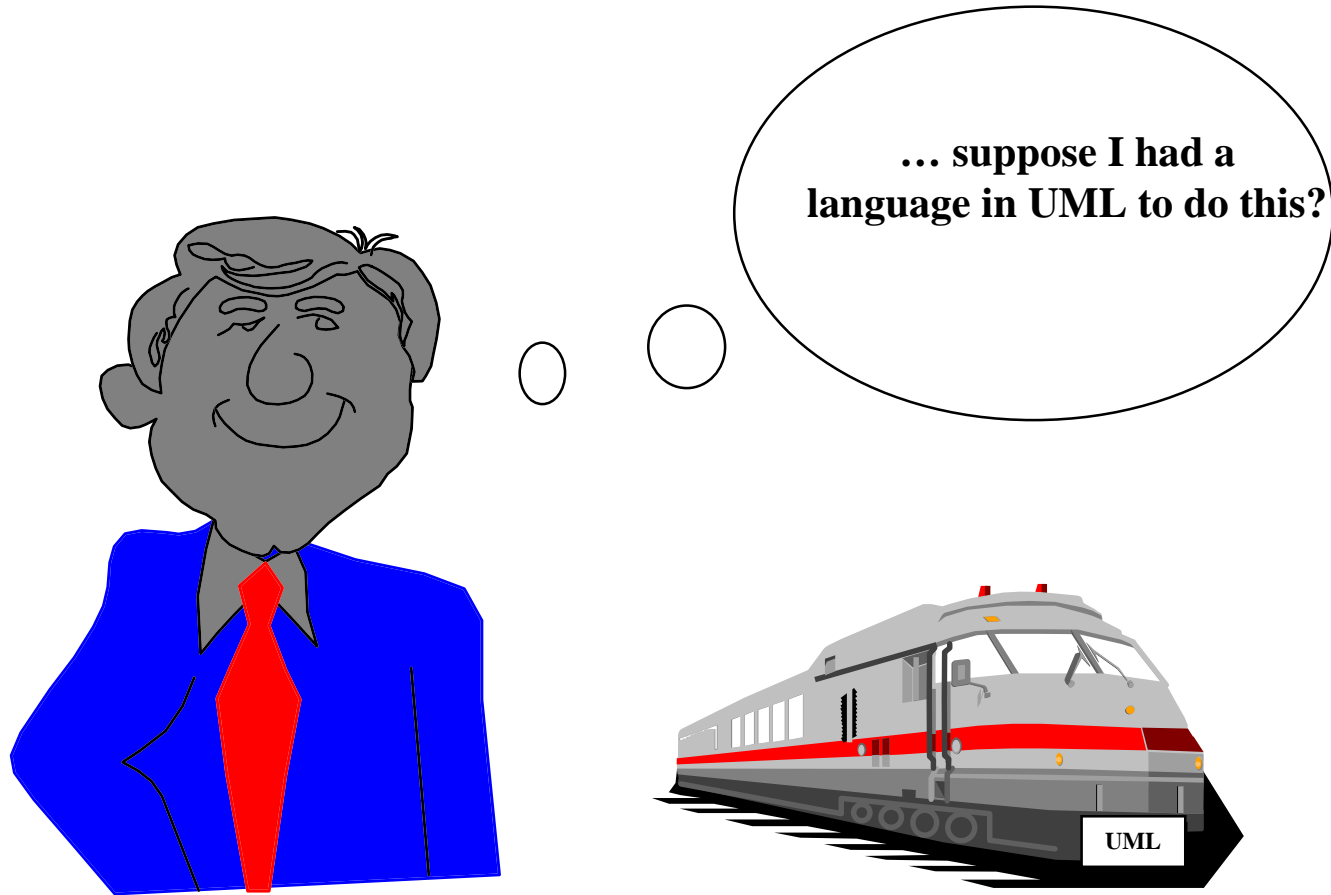
... but how do I specify  
the details of my  
actions and operations?

I need this to  
verify my models and  
translate them into code!



# New Possibilities Arise

---

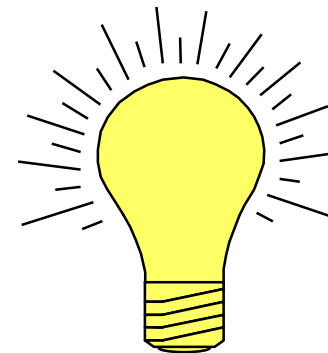


# Action Language

---

An action language is a *specification* language

- \* to specify *actions* and *activities*  
on a *state diagram*
- \* and *operations*  
on a *class diagram*



# Action Language Goals

---

An action language should meet the following goals:

- industry standard
- *complete* processing specification
- verifiable through simulation
- 100% code generation from UML models

# Basic Characteristics Required

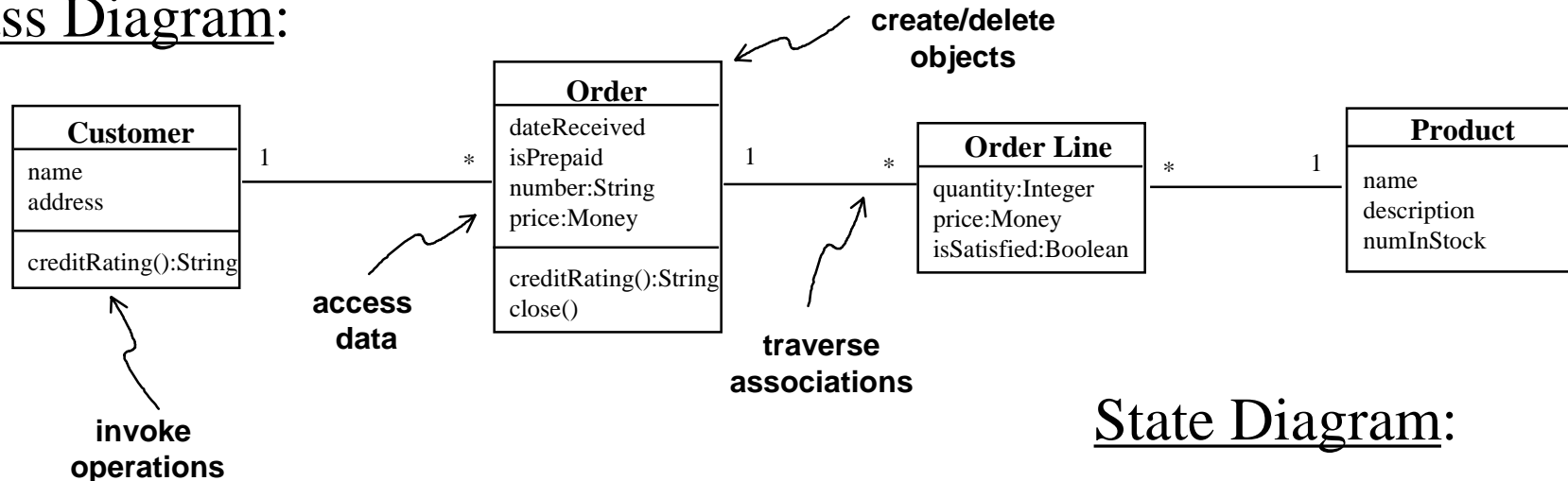
---

An Action Language would need to be:

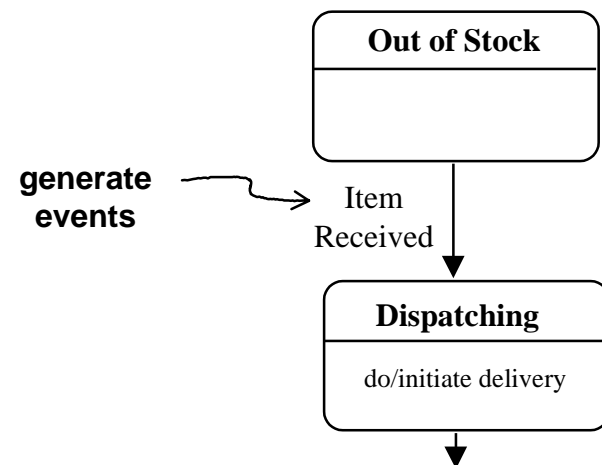
- UML compatible
- executable and complete
- implementation independent
- at a level of abstraction above implementation

# UML Compatible

## Class Diagram:



## State Diagram:



Must support interactions with defined UML elements

# Executable and Complete

---

## Real Computing on Real Data

Traversing Associations

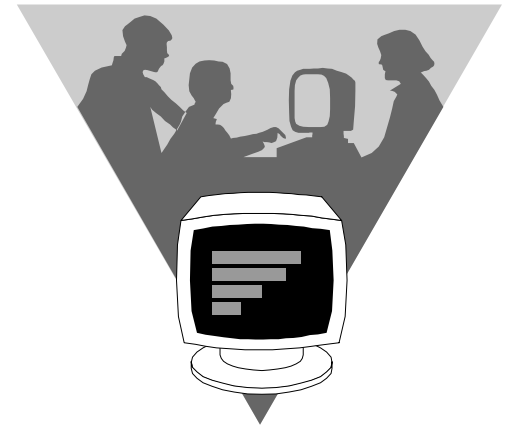
Data Access

Signaling Events

Computations

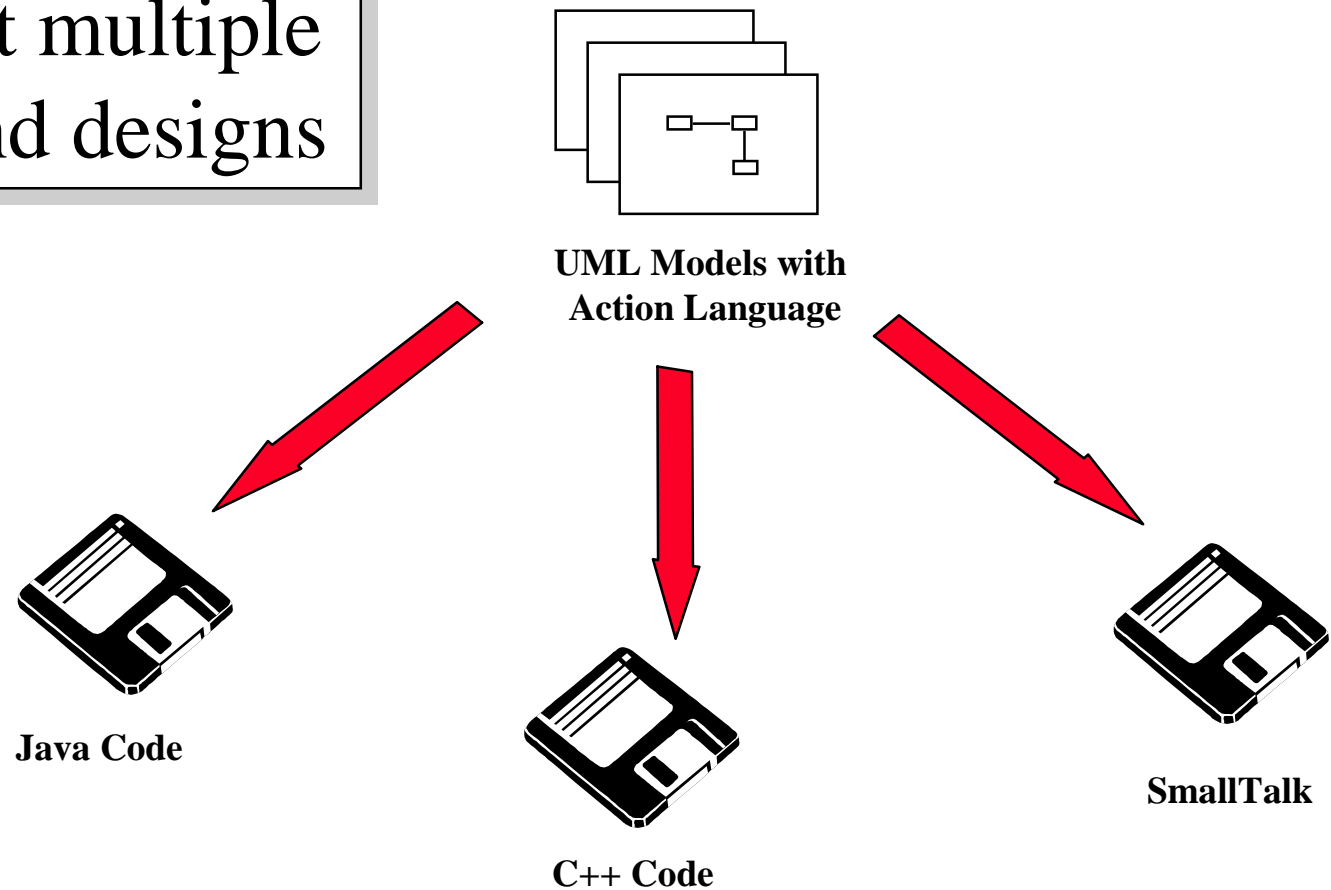
Creating and Deleting  
Objects

Creating and Deleting  
Associations



# Implementation Independent

Must support multiple languages and designs



# Higher Level of Abstraction

---

## Greater Productivity:

Engineer must be able to express specifications without attending to all of the implementation details.

## Implementation Flexibility:

Automated translation must be able to optimize the generated code, possibly by *reorganizing the structure of the model*.

# Language Considerations

---

## Atomic Association Traversal:

```
self ->[Order->]Customer.name
```

Traverses from the current instance of Order Line across two associations to access the name attribute of Customer

## Separate Access and Computation:

```
(self -> Product.numInStock, self.quantity) |  
diff > self -> Product.numInStock
```

Traverse from a current instance of Order Line to Product to access the numInStock information

Computation

# Language Considerations

---

## Collections:

```
self -> [Order->]OrderLine.price
```

Traverses from the current instance of Customer to Orders (plural) to Order Lines (plural) and accesses the resulting collection of prices.

```
self -> [Order->]OrderLine.price | max  
..
```

Computations can be done on collections as well.

## Order of Computation:

```
Do_this; Do_that;
```

These could be in either order, or even execute concurrently if there are no dependencies

# UML Action Language Benefits

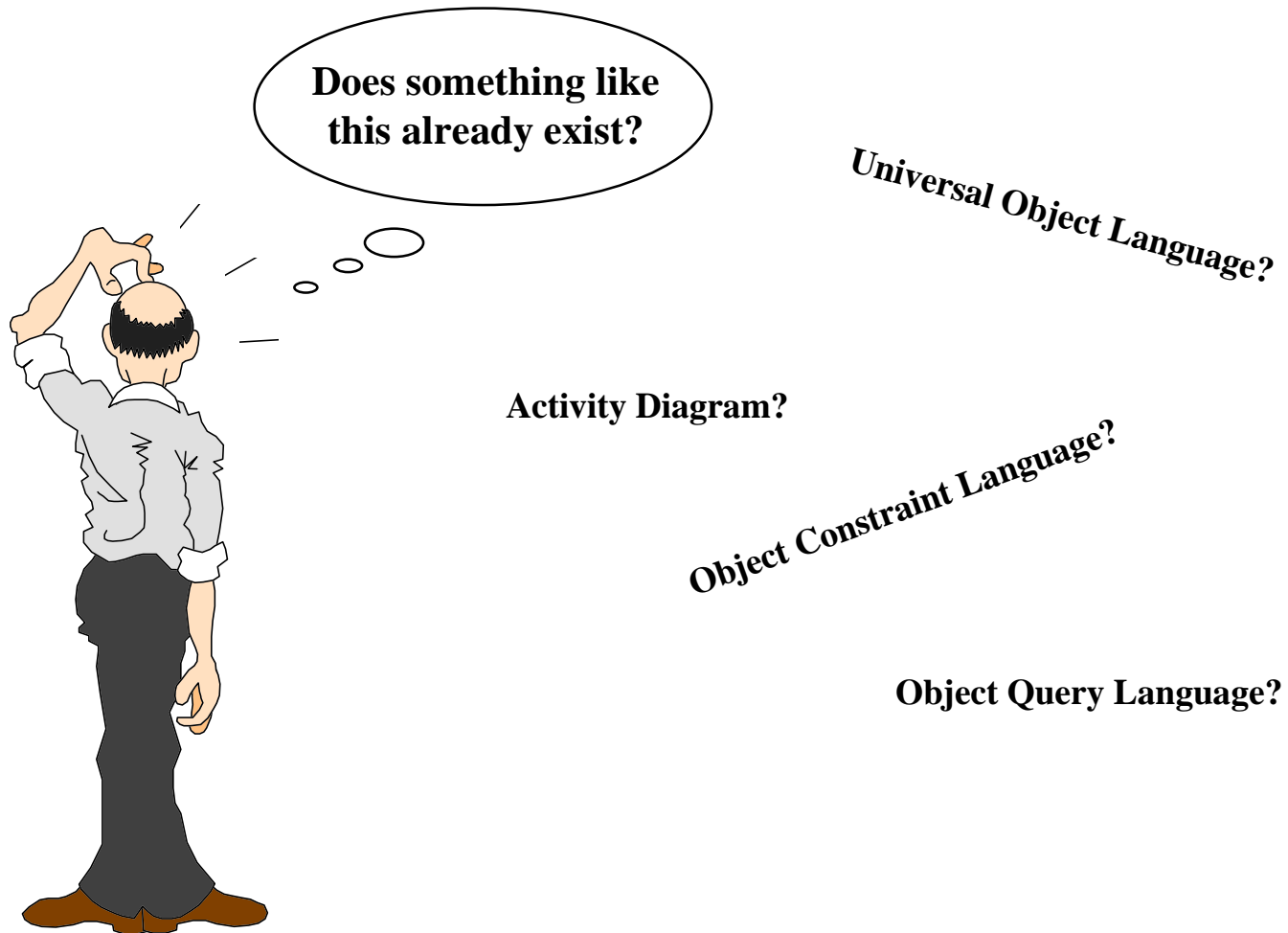
---

An action language would provide the following benefits:

- **Inclusive** - support both elaborative and translative approaches
- **Standard** - extend a valuable industry standard
- **Identify Specification Defects** - earlier, lower cost
- **Increased Productivity** - 100% code generation from UML

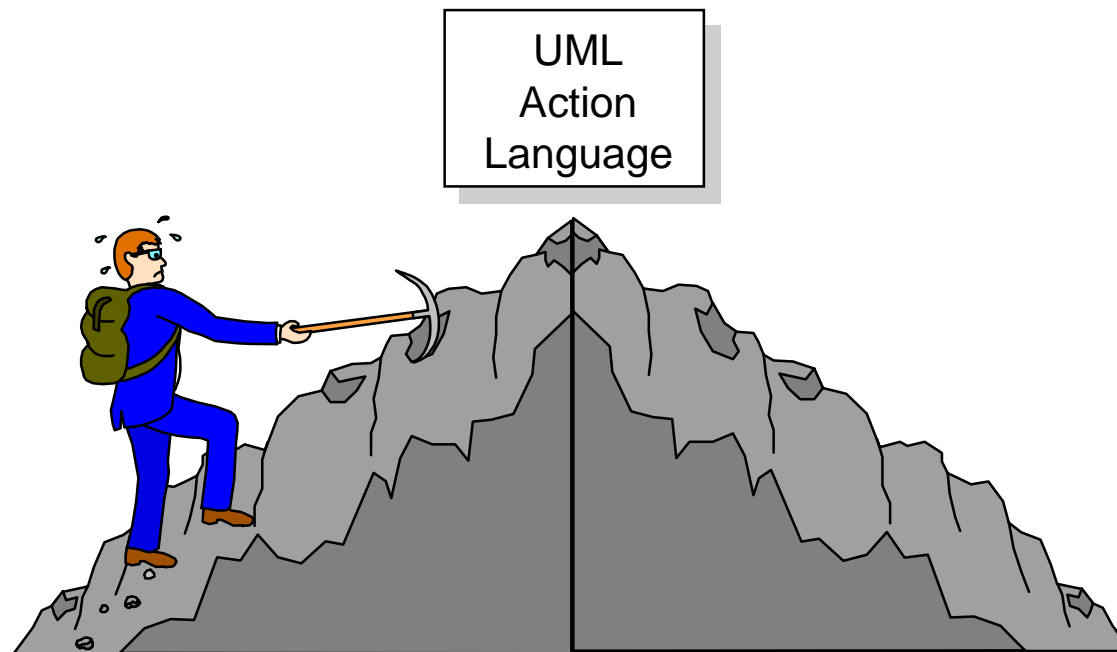
# Existing Languages and Standards

---



# Something More Required

---



We believe the OMG should consider a RFP  
for a UML Action Language