Executable Modeling: Retrospective and Prospective

Stephen J Mellor
A short history of MDD

Structured Design: Yourdon and Constantine 1979
Structured Analysis: De Marco 1981
OOA: Shlaer and Mellor 1988
OO Design: Booch 1988
OMT: Rumbaugh et al 1992
Object Lifecycles: Shlaer and Mellor
Structured Devpt/RT: Ward and Mellor 1985
UML 1.1: Three Amigos 1997
Executable UML: Mellor and Balcer 2002
UML 2.0: Cast of thousands 2005
Types

In the early Nineties, we received a fax from the OMG requesting participation in a Unified *Method*.

---

In the early Nineties, we received a fax from OMG requesting participation in a Unified Method.

**Big Mistake**


Rodney Bell
Two forces

Modeling formalism should be close to the knowledge we’re capturing.

Modeling formalism should be close to the implementation.
“The Unified Modeling Language is a language for specifying, constructing, visualizing, and documenting the artifacts of a software-intensive system.”

The UML Summary

UML is not a method

® Object Management Group
Agile Manifesto

Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.
The reason code is so important is that it runs, right?

An executable model runs, so it can be verified, right?

So if a model can be executed, it is as good as code, right?

Argh!!!!!!!!

Yes!

Yes…

No. Code is the most important thing.
Action Language

- Add code progressively
- Add “model-aware” code progressively
- Add traditional model-aware code
- Add model-based code and translate it
Action Language

Elaborative

Structural

Behavioral

Executable

Translative

Executable and Translative
Elaborative development

Analysis → Preliminary Design → Detailed Design → Target Code

- Design and Architecture Details
- Implementation Details and Code Bodies
- Code Generation

Intermixed Application and Design
Manually Created Code Bodies and Implementation Details Required for Model Execution and Code Generation
Target Code assembled from Hand-Coded Bodies inserted into a generated framework
Change the assumptions….

- Fully detailed analysis
- Translator
- Design and architecture details
- Code
“Recursive Design”

Recursive Design views system design as

- a process of *systematic translation* of
- an application
- according to a set of rules
- into code.
Recursive Design...

Physical Telephone

On Hook

Off Hook

Application Model

Generator

an application

systematic translation of

Model Compiler

a set of rules

into code.
Uniformity

A minimal, uniform set of organization rules:

- reduces cost of understanding, building, and maintaining the software
- decreases integration effort
- leads to smaller, more robust code

This uniform set of organization rules is a software architecture.
Multiple Architectures

There may be several implementations with different performance properties.
Separation of Subject Matter

A system comprises several different subject matters.

Recursive Design externalizes dependencies.
Domains

Each subject matter is a problem domain.

Bank
- Customer
- Account
- Transaction

AUI
- Screen
- Selection
- Text Line

Each domain has its own vocabulary.
Composable Domains

Model each domain, then connect them.

Bank

Class Diagram

Customer

1

1...*

Account

State Chart and Operations

Window

Button
The Industrial Internet

It’s an internet of things, machines, computers and people, enabling intelligent industrial operations using advanced data analytics for transformational business outcomes.
The Future

“Industrial Internet of Things: Unleashing the Potential of Connected Products and Services” by the World Economic Forum, with Accenture

1. Operational Efficiency
   - Make specific operations more efficient:
     - Asset utilization
     - Operation cost reduction
     - Worker productivity

2. New Products and Services
   - Shift from selling assets to selling outcomes:
     - Pay per use
     - Software-based services
     - Data monetization

3. Outcome Economy
   - Shift from selling assets to selling outcomes:
     - Pay per outcome
     - Connected ecosystems
     - Platform-enabled marketplace

4. Autonomous, pull economy
   - Continuous demand-sensing
   - End-to-end automation
   - Resource optimization and waste reduction
When will be Executable Models be Commonplace?

1985: “In three years time…”
1987: “In three years time…”
1989: “In three years time…”
1991: “In three years time…”
1993: “In three years time…”
1995: “In three years time…”
1997: “In three years time…”
1999: “In three years time…”
2001: “In three years time…”
2003: “In three years time…”
Thank you