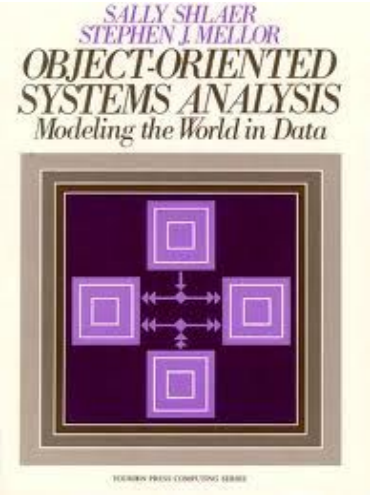
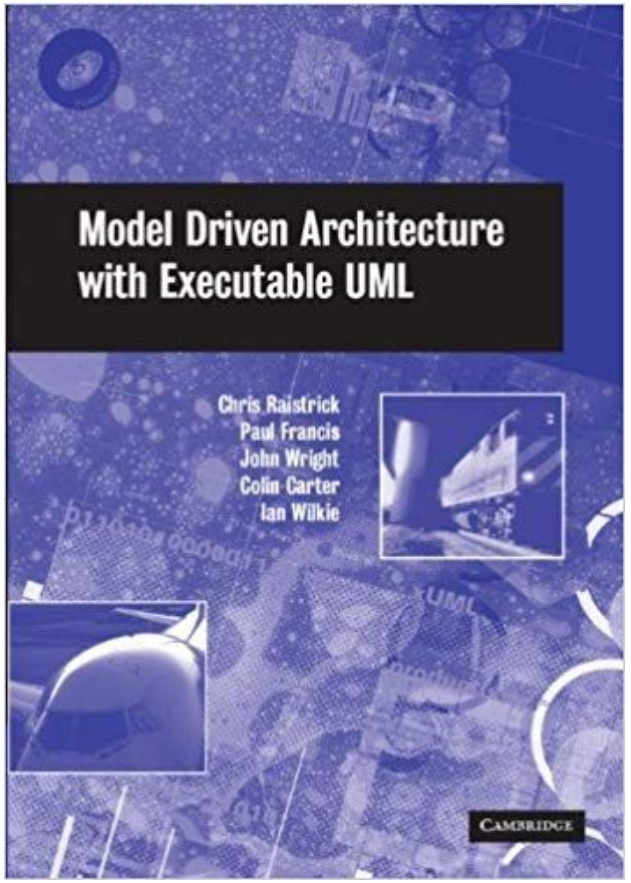
OOA '20
Overview and Panel Debate

Agenda

- OOA '20 Report?
- Motivation
- Outline
- Method Debate
- Call for Collaboration





OOA '20 Report

The Shlaer-Mellor Method was published starting in 1988 and 1991 with *Modeling the World in Data* and *Object Lifecycles*. It has been refined through the years in follow-on books and reports. During the most recent decade, little has been published in literature while the tooling and practice has continued to evolve and extend the Method by default. Tooling and practice has made choices in corners of the Method not clearly defined in literature.

It is time for a publication that highlights these extensions and details taking on a position on best practice for the Shlaer-Mellor Method. Such a publication will require collaboration among currently active modelers and tool smiths.

Motivation

- Clarification of the Method
  - Several elements are used in tooling but have never been described in the literature. It is time to catch up.
- Formalization of the Method in xtUML
- Consolidation and broadening of the community
- Seeding interest in a wider range of audiences (including academic audiences)

Outline: Part 1 - Method Report

System modeling

- Components
  - Component cardinality
- Interfaces
- Ports
  - Polymorphic port
- Messages
  - Synchronous/asynchronous
  - Returning values
  - Mapping to class-based operations or functions
- Deployments

Outline: Part 1 - Method...

- Dispatch
  - simultaneous expired timers (delayed events)
  - event queue overrun (Consider 'simulated time'.)
  - Delayed accelerated event to self
  - Message dispatch
- Class-based state machines
- Creation events
- Type system
  - Collection types

Outline: Part 1 - Method...

- Packaging
  - Package references
  - “Domain” packages
  - Packaging idioms and conventions
- Error handling
  - Cannot happen
  - Exceptions
  - User defined exceptions

Outline: Part 1 - Method...

- Action Language Dialects
  - OAL
  - MASL
  - ASL
  - Alf
- Mellor-Balcer/Raistrick rationalization:
  - Polymorphic event
  - Baseless referentials
    - Referential as identifier behavior
  - Final states
- xtUML relationship with UML (as specified by OMG)

Outline: Part 2 - Tooling Survey

- BridgePoint
  - Commentary on BridgePoint deviations from the method
- iUML
  - TBD
- MASL architecture
  - TBD
- inspector
- Others

Method Debate

- System Modeling
- Event Dispatch
- Class-based State Machines
- Creation Events
- Type System

Method: System Modeling

(Inter-domain) Messages

synchronous and asynchronous

Should there be a distinction between *synchronous* and *asynchronous* messages between domains? If so, what are the semantic differences?

synchronous

Should senders block on transmission of synchronous inter-domain messages returning no data (void)?

[Scenario] Consider domains 'navigation' and 'braking'. A statement within the the interior of a state action of an instance state machine in 'navigation' sends the 'apply friction' message (invokes a terminator service) on 'braking'.

See *Action Semantics for the UML* for discussion on intra-domain messaging semantics.

Method: System Modeling

Domain Multiplicity

"instances" of domains

Can a domain be reused multiple times in the same system? If so, how are messages directed to the correct instance of the domain?

[Scenario] Consider domains 'navigation' and 'braking'. A statement within the the interior of a state action of an instance state machine in 'navigation' sends the 'apply friction' message to the "left rear instance/instantiation" of the 'braking' domain.

Method: Event Dispatch (1)

delayed events

simultaneous expired timers (delayed events)

How should simultaneously expired delayed events be dispatched?

[Scenario] Consider a domain with empty event queues and 3 pending timers. All 3 delayed events are past their expiration time. The first 2 delayed events in the queue have the same expiration time.

Method: Event Dispatch (2)

event queue overrun

dispatch of delayed events in the face of full queues

Should delayed events be dispatched when expired without regard to the presence of non-delayed events?

[Scenario] Consider a domain with non-empty event queues and pending timers. At least one delayed event is past its expiration time.

[Edge Case] Consider a cyclic, "self-stimulating" state machine (or interacting set of state machines) that spins with no delay.

The concept of 'simulated time' or 'Discrete Event Simulation' may inform this discussion.

Method: Event Dispatch (3)

Delayed accelerated event to self

accelerating delayed events

Is a delayed event to self treated the same as a delayed event to any instance?

[Scenario] An instance state machine sets a timer and supplies an event directed to itself.

Method: Class-based State Machines

- **Are general class-based state machines a *Good Thing* and to be supported in xtUML as today?**
- **Should class-based state machines be limited to associative classes only?**
- **Should class-based state machines be deprecated?**

Method: Creation Events

- **Are creation events a Good Thing and to be supported in xtUML as today?**
- **Should creation events be deprecated? Synchronous creation is sufficient.**

Method: Type System

- Collection types
  - **Is the existing Type System (of classes, user data types, enumerations and structures) sufficient?**
  - **Are collection types (arrays, sequences, bags, sets, dictionaries) missing from the Method?**

Method: Polymorphic Events

- **Should Polymorphic events be "concrete" visiting each level of the hierarchy? (K-C)**
- **Should Polymorphic events be "abstract" consumed at exactly one level of the hierarchy? (PT)**
- **Should the above be consistent with sub/super (deferred) class operations?**
- **How shall the Method define a coexistence of concrete and abstract polymorphic events?**

Call for Collaboration

Call for Collaboration

- contributing authors
- editors
- reviewers
- financial underwriters

OOA '20 Report