MODeX
Model-Oriented Data eXchange

Challenge: Create a domain-specific modeling language for enterprise data integration

MODeX is a model-driven solution for Enterprise SOA. Where general-purpose SOA and MDD toolsets focus on developer productivity, MODeX is additionally concerned with enterprise-scale issues like versioning, standardization, and reducing long-term barriers to system integration.

Versioning

Minor versions can add optional fields. Routines support minor version bridging, and unknown field data pass-through.

Major versions can completely alter fields: add, remove, delete, change optional/mandatory status. Does not work with existing contracts, but tooling can allow smoother major version migration.

Modeling tool maintains multiple copies of Entity for every version, minor or major.

Data Aspects

- A way of specifying cross-cutting concerns in messaging
- Allows introduction of features without changing the underlying entity model
- Extensible to allow new aspects to be woven in as necessary

Rich Validation Rules

- Ability to extend validation beyond what schema allows.
  - Templates for common validation patterns:
    - If Field 1 = X, then Field 2 must = A, B or C
    - If Field 2 is populated, Fields 2 and 3 must also be populated.
  - Integration with JBoss Rules for open-ended expression of complex rules

Specifications

Core Technologies

Full-Fidelity Messaging

Federated Model Repository

Rich Validation Rules

Data Store Integration

MODeX Designer (Graphical Modeling IDE)
Model/Schema Utilities (Process/Assembly, Build, Publish, etc.)

CodeGen Generator

Schema Targets (Object Pool, Text Form)
GDA Targets (Java, C++, C#, .NET)

Runtime Support Libraries
Marshalling, Version Bridging, Validation, Encryption...
Java Runtime
C# Runtime
C++ Runtime
Other languages...

Specifications

Building blocks for message contracts
- Specified in terms of the underlying entity model
- Allows presence constraints: optional, required, excluded
- Extensible to allow additional constraints

Message Contracts

Message Structure

Data Security
- Allows presence constraints: optional, required, excluded
- Extensible to allow additional constraints

Full-Fidelity Messaging

Rich Validation Rules

Federated Model Repository

Rich Validation Rules

Data Store Integration

- Semi-Static Enumerations: a primitive field whose values are validated against a database or data service.
- Enrichment On Demand: dynamic lookup and population of values as object graph is traversed, and as values are requested

Full-Fidelity Messaging

- Marshalling will reconstruct object graph with full fidelity.
- Compilation time will evaluate optional/mandatory fields.
- Marshalling will pass through unknown fields.
- Marshalling will support minor version bridging.
- Marshalling will support major version migration.

Full-Fidelity Messaging

- Marshalling will reconstruct object graph with full fidelity.
- Marshalling will pass through unknown fields.
- Marshalling will support minor version bridging.
- Marshalling will support major version migration.

Full-Fidelity Messaging

- Marshalling will reconstruct object graph with full fidelity.
- Marshalling will pass through unknown fields.
- Marshalling will support minor version bridging.
- Marshalling will support major version migration.