MDM. Provenance. Versioning. Dimensional context values / queries. Model Facets APIs: Functional, Semiotic, Dimensional Dataflow contexts / order / roles HATEOAS APIs workflows. Protocol: Dialog. Browse / analyze / transform "activations" (REST / JAF) according Facets. Clients / Connectors.

Contents / Features (Mision / Vision). Distributed consistent Knowledge Applications. Trust. Consistency. Event sourcing. Inferencing (of distributed state). Reconciliation.

Certify distributed Entity / Subject Identity / State (in roles / dimensional points). Class / instance alignment (matching).

Integration: Augment sources / back ends. Model I/O materialized in source (plugged) application / services back ends.

Integration: Extension. Extended functionalities data / schema / behavior exposed as services external to source (plugged) applications. Sync (Augment). Declaratively stated via Model descriptions. Discoverable, browseable (HAL / REST).

The idea of the project is to "augment" an ESB for EAI platform and to enable it allowing it to make "inferences" regarding which routes to use, "discovering" sources / destinations of an event message(s) which then it transforms / enriches according destination "semantics" and format(s).

This featuring the exposure of a generic facade which allows to see in an "homologated" view the applications or services and their data, schema and behavior (actions) that could be integrated into the tool.

Different integrated applications are enriched with this facade and with the events that, given the inferred routes and transformations, augments theirs data, schema and behaviors, invoking activities corresponding to each destiny semantics.

Resource monad of Model(s) layers Contexts: Transform, Kind, Entity, Behavior, etc. Model(s) Context hierarchy classes (model) instances (domain). Contexts, Occurrences, Attributes, Values : CSPO Functors (eval in Resource contexts). Functor declarations (model augmentations / domain dataflows encoding).

Monads / Functors examples (order: comparables / upper / dimensional ontologies):

Kind<SubjectClass<Resource>, Set<PredicateClass<Resource>>>, others. Example: Subject / Predicate(s). (quads prev / next relation).

Metaclass<ObjectClass<Resource>, Set<OccurrenceClass<Resource>>>, others. Example: Object / Statement(s). (layers parent / child occurrences relation).

Class<ObjectClass<Resource>, Set<ContextClass<Resource>>>, others. Example: Resource / Kind(s) (class definition relations: extension / intension, layers parent occurrences prev relation).

Backend: RDF(S), OWL, Resource Services (Naming, Index Registry), ISO, Shapes, IDs matching. DIDs Backend.

Runtime: Objects, Events. Distributed Runtime Functional (Dataflow) reactive Resource objects / models (forms / flows).

Deployment: Spring / Vertx / others (Jersey / CDI: Resources Meta Model / Domain APIs). HATEOAS / HAL APIs (Encoding / Dialog Endpoint Protocol). Environment / Container: Messaging / Services. Models I/O.

Integration Connectors: Runtime Model embedded Resources. Sources as reactive stream objects. API.

Integration Clients: Runtime Model embedded Resources. Services as reactive stream objects. API.

Integrations (augment / extend): RDBMS (R2QL / R2ML) / Rules / BPM, KIE / Teiid / Metamodel / Olingo / Solid / ERP / CMS / Feeds (Hypermedia Resources / Apache Stanbol) / Browser / Messaging (JMS overlay, Apache Camel / ActiveMQ: ServiceMix Connectors / Clients) / ML, Big Data (Apache Spark).

Dialog: Encoding / Protocol. Activation, Location, Context (Hypermedia type / context interfaces). Augmentations.

Encoding: XML / XSL / XPath / XLink / XForm / XPointer / XQuery (RDFS / OWL / ISO DM / RM Forms / Functors / DOM). Endpoints (streams / signatures): documents (messages) / resources (paths).

Comparable: parent / child relations (SortedSet). Model sets / dimensional alignments.

Models: Augmentation, Kind (Metamodel), Entity, Class, Behavior (Functional), Dimensional, Semiotic (Matching / Alignment / Upper).

Runtime. Bus. Signatures bindings (reactive), content type / domain context interactions data transforms (roles).

Design: Augmentation (Aggregation, Alignment, Activation Functors), Domain Functors. Encoding. Design Functors Dataflows.

CQRS. Event Sourcing. Functor Commands.

Protocol / Encoding: hierarchical contexts dialog (runat) prompt / pick / select (roles).

Endpoints: Model (classes) objects / resources, Domain (instances) objects / resources

Apache Isis (DDD / Actors). Connector / Client / Runtime (Hypermedia / Facade / Dialog Protocol: services / actions). Backend: Augmented Services / Persistence. Representations (resource: types / activation). Domain, Services, View Models.

JBoss Teiid / Apache Metamodel: Clients / Connectors / Backend / Runtime (Translators).

CMS / Wiki API / WebDAV. Apache Stanbol. DOM / REST / JAF.

StratML Client / Connector (Goods, Needs, Products, Goals, Purposes. Exchange ontologies).

Ontology Matching: Dimensional metaclass / class / instance attributes relationships in axes / contexts / roles evaluations. Resolve equivalence of dimensions, units, measures, values via aggregation of value occurrences. Resolve value kinds by occurrences of Semiotic (grammar) model. Contexts (CSPOs) Attribute / Value "clustering". Sets specifications. Dimensional order relations.

Distributed Inference (dimensional / matching / predictions). Distributed consistency. Event sourcing (Dialog, CQRS Protocol). HATEOAS distributed / lazy reconciliation: hashing of source / destination (hierarchical contextual model quads hashing) state in interactions tokens. Prompts. Distributed Alignment (until reach of goal synchronization states). Distributed state / IDs.

Comparable: parent / child relations (SortedSet). Model sets / dimensional axes alignments.

Semiotic: syntax, semantics (grammar), pragmatics. Verbs: action, passion, state (for / due to action / passion). Models. Dimensional (axis / order) relations. Alignments. (Sorted) Set relations: hierarchical encoding. Verb roles (CSPO / Kinds sets).

Example aggregation: candy (type), red (color attribute), strawberry (flavor attribute).

Encode IDs: Context Kind, upper (meta) Resources (levels / layers). Resource contents / contexts (identify by occurrences in roles in other contexts, Meta Resources, layers class, metaclass, instance). Compose IDs (hierarchical graph properties encoded string) from outer to inner resources (Context, Kind, Occurrence, Role, Resource). "Operable" IDs (ClassIDs / InstanceIDs: Meta Model reifications / occurrences).