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).

Message parsing (Template, Message Context) matches Form / Flow Augmentation Mappings signatures.

(Template, Context, Attribute, Value); Value as Context: hierarchical models. Same attributes: types / collections.

Semiotic (encodings):

(Context, Sign, Concept, Object);

Object as Sign: Object properties (Concepts).

Semiotic encoding: encode Meta Model (syntax facet) / Functional (Meta Model encoded pragmatic facet) / Dimensional (Meta Model encoded semantic facet) data / reference model (by contexts / upper alignments / aggregation).

Facets contexts semiotic encoding. Facets IO (events) by semiotic encoding of facets input layers.

(Context, Sign, Concept, Object);

Roles encoding: Object as Sign (properties), Sign as Object (types / roles), Concept as Sign / Object / Context, etc.

Grammars encoding: express models contexts layers and augmentation templates in input contexts.

Meta Model:

(Transform, ID, ID, ID);

(Mapping, Transform, ID, ID);

(Template, Mapping, Transform, ID);

(Augmentation, Template, Mapping, Transform);

(Resource, Augmentation, Template, Mapping);

(Role, Resource, Augmentation, Template);

(Statement, Role, Resource, Augmentation);

(Model, Statement, Role, Resource); Resource Occurrence in Model (Kind? Kind Role type, Kind hierarchies.)

Functional Facet:

Occurrence

Entity

Kind

Class

Flow

Behavior

Dimensional Facet:

(Value, Previous, Distance, Next);

(Measure, Value, Previous, Distance);

(Unit, Measure, Value, Previous);

(Dimension, Unit, Measure, Value);

(Concept, Dimension, Unit, Measure);

DCI / DOM: Subject, Context, Occurrences, Roles, Attributes, Values, Activation.

Semiotic, Meta Model, Dimensional, Functional DOM. DOM Contexts. DOM Functors. Resources Contexts (CSPO Monads: encoding / addressing).

Activable DOM Contexts: DOM views (object mappings, etc.).

Map Reduce encoding inputs, grammar templates context mappings. Emit Semiotic reference / data model parent / child properties encoding (Context / SPOs, Context:Subject / POs, etc.).

Model:

Integration: Purpose (description, mision, vision).

Task list focus for each item.

Analysis (mision).

Design (vision).

Containers / Components / Addressing / Routing / Protocols / Interfaces / Services (Message IO data / Backend Models schema / Behavior encoding).

Deployments (use case) Goals app (tasks, exchanges, etc. over integrated backends: "wizards").

Model / Semiotic reification.

Facets / Levels / Shapes: Aligned entities / values occurring in aligned models / dimensions.

(Context, Sign, Concept, Object);

Resource Monad / Message Functor (Contexts hierarchy parent). Resource<Context>. Events bindings. Message / Augmentation event declarations / instances.

Resource

(Message, LHS, Cond, RHS);

(Context, Sign : Message, Concept : OntResource, Object : ID Resource);

(Transform, Context, Message, Concept);

(Mapping, Transform, Context, Message);

(Template, Mapping, Transform, Context);

(Augmentation, Template, Mapping, Transform);

(Resource, Augmentation, Template, Mapping);

(Role, Resource, Augmentation, Template);

(Statement, Role, Resource, Augmentation); Augmentation of which Statement is result of.

(Model, Statement, Role, Resource);

(Entity, Model, Statement, Role); Model (Backends) aligned entities.

(Kind, Entity, Model, Statement);

(Class, Kind, Entity, Model);

(Flow, Class, Kind, Entity);

(Behavior, Flow, Class, Kind);

(Measure, Behavior, Flow, Class);

(Value, Measure, Behavior, Flow);

(Unit, Value, Measure, Behavior);

(Dimension, Unit, Value, Measure);

Functional: Entities in different Models in Kind, Class, Flow, Behavior.

Dimensional: Measures in different Dimensions in Units, Values.

Value, Previous, Distance / Event, Next. Order (in axis):

Siblings: previous / next (Semiotic containment relationship / roles). To do.

Hierarchies: parent / children (Semiotic containment relationship / roles). To do.

Semiotic model: reification / primitives / upper ontology (meta model, inferred / aggregated). Levels: syntax / grammars, semantics, pragmatics. To do.

Model (meta / upper resources) / DOM (domain instances) Augmentation Functors:

Aggregation: Create. Context occurrences. Materialise into next layer(s) streams. Context occurrences statement(s) (children: next layers). To do.

Alignment: Update / Retrieve. Augment attributes / values. Materialise into next context augmented statement streams (next context occurrence attributes / values, layer siblings). To do.

Activation: type / role in contexts. Parent (occurrence Context) occurrence Kind for Context object (parents: previous layers). To do.

Activation: type / role in contexts. Parent layer (occurrence context layer) CSPOs Kind(s). Materialise into dataflow Message(s) streams: (CK, SK, PK, OK). (parents: previous layers). To do.

Example:

Entity in (Model, Statement, Role, Resource).

Kind in (Entity, Model, Statement, Role).

Aggregation / Alignment Messages from primitives / meta / upper / domains resources Activation.

CSPO: Context (intension), Object (extension). SortedSet comparison hierarchies (encode order in semiotic DM / RM, octal comparison values).

Message: Augmentation (materialised Resource set) specification (functor). Event declaration (I/O patterns bindings). Context layers monad publishes / subscribes (dataflow bindings pipelines).

Addressing (hypermedia):

Content Type: Metaclass

Resource: Class

Representation: Instance

Address: Occurrence / Response (Materialised Message Augmentation)

Request: Message (state flow). Context DOM event API: Request Message Resource in possible domain / range / mapping contexts.

URIs: Contexts (data), Types (reified schema), Service (behavior). Class, instance, occurrence links, paths, pointers (annotations / transforms / embeddings / calculated / descriptions).

Persistence: distributed event queue (saga) of distributed addresses (occurrences metaclass, class, instance resolution). Composite occurrences (links / rels / roles).

Purpose / content type driven (state / rendering / roles / links /contexts / data / flows / attributes) declarative hypermedia activation application. Model, Application, Domain ontology / upper resources (connectors). Generic API / metamodel (DCI: Form / Flow) client. Extension protocols / APIs.

Resource: Model, graph (encoding). Dataflow: order encoding. Sort dimensional measures (SortedSet?).

Encoding: Semiotic levels metaclass, class, instance, occurrence (in context in role: metaclass?).

Ontology matching: signatures encoding. Dimensional ordered aggregated measures. Sets. Semiotic reification.

Use Case: Goals App

Goals App: purpose / goals / domain driven syndication of integrated business / social / cloud application features. User / Groups / Roles Purpose(s), Goal(s), Task(s) "intelligent" tracking oriented focus providing an abstraction and integration layer of players process flows / interactions and players process assets management and semantic orchestration.

Goals App: Semantically annotated gestures / interactions (contexts, purposes messages / interactions / resources / content). Subject context occurrence role attributes values (metaclass, class, instance, occurrences).

Goals App: API Facade for rendering aggregated data roles in contexts interactions topics / subjects assets (conceptual domain contexts axis / state views / activations: Forms / Flows). Example: domain declared Customer (actor / role), Product, Order, Purchase, Invoice, etc. topics / subjects assets rendered in contexts (Sales Report, Expenses Report, etc. embedded / linked dashboards). Wizards.

Goals App: Browse / search / activate: history / relations / referrer context / interaction / gestures roles traceability / (dialogs). Gestures / interactions (actor / asset, actor / actor). Wizards.

Goals App: Hypermedia contents APIs (embedded / embeddable resources: Semantic contextual Wiki / Apache Stanbol / CMS: hypermedia augmentation, knowledge / behavior maps). Integration: augmentation / sync backends / apps. Extension: services / APIs. Annotate / augment link content. DAV protocol (integration / extension facades).

Low level Resource / Message / Context model / layers API. REST. Render DOM Context / OGM Domain (model) instances: Restful Objects / Apache Isis / HAL / GraphQL (meta / domains models endpoints) like APIs. Forms / Flows MVC / DCI APIs (connectors / clients / adapters).

Transactions: Request IDs. CQRS. Reified interactions / gestures. Event sourcing / logs. Reified (resource) Message activation (dataflow) Forms / Flows.

Reference Model / Encoding:

Syntax: data (signs). Semantics: schema (concepts). Pragmatics: behaviors (objects CRUD / Interactions / Contexts).

Reify able roles. Agregate semiotic Activation of a statement resources roles. Example: aggregate kinds / entities. Instance (extension) / Class (intension): Object / Concept.

Order / hierarchies: reified Context / Concept roles relation (example: Concept as Context, Concept aggregations are siblings / children of first Concept).

CSPO Reference Model: (attribute / value). Reified roles (Semiotic layer). Order. Hierarchies.

Encoding: To do. Compatible with Message Augmentation Mappings Predicate comparison and Index traversal.

Index: To do. Compatible with Message Augmentation Mappings Predicate comparison. Encoding representation (cons list, graph, tree, etc.). Truth values (flows) traversal.

Comparison: Ternary truth values results (previous / parent, current / OK, next / children) according Predicates for a Message resource set and and a Mapping comparison (apply augmentation).

Predicates: quad encoded comparison specification, result of iterating Mapping comparisons until Augmentation Message matches all Resource set. Augmentation comparison behavior. Message / Augmentation application specification.

Mappings: comparison iterations matching Message / Augmentation Predicates patterns truth values. Compare input Message resource set with Mappings till truth values are all OK. Perform Augmentation.

Model encoded Augmentation (Aggregation, Alignment, Activation) determined Comparison, Predicates, Mappings.

Encoding: IDs. Relative / contextual metadata. Serialization. Index.

Event: Message (specification), Augmentation (instance).

Predicate: comparison specification (axis). Message SPO.

(Message, LHS, Predicate, RHS);

Mapping: comparison result (patterns). Augmentation SPO. Template Transform rendered in Message context resources set.

Abstract Adapter / Connector / Client APIs:

Populate Model Context layer (Model Statements).

Protocol: Form / Flow Context (interaction). I/O data / schema / behavior input / output abstraction. Domain (model driven) interfaces / schema.

Inputs (data): Resources augmented / aggregated from OntResource until Dimensional layer.

Protocol (schema / behavior): Browse from Dimensional up to OntResource layers selecting, creating and matching intermediate Context(s). Form / Flow interaction use case roles.

Aggregation / augmentation: feedback from interactions (levels) CUD interactions.

Adapter Meta Resources (upper). Translate inputs. Render outputs from Model protocol into backends adapter specific protocol APIs.

Languaje levels: Use, mention. Pronouns.

Sign, Interpreter, Concept, Object: relations (syntax, semantics, pragmatics).

Adapter Model: metaclass, class, instance, occurrence. Context (interpreter), Sign, Concept, Object.

Connector I/O (gestures):

Client I/O (dialog):

Model layers:

OntResource: Resolves reified aligned / matched aggregated Resources.

Predicate: 'kind', aggregates roles attributes / values. Grammar.

Encoding: metaclass, class, instance, occurrence (contextual / order/ ops) IDs.

Encoding: Sets CSPO Context specification (sets quad encoding).

Model:

(OntResource, OntResource, OntResource, OntResource);

OntResource represents aggregated / matched different identifiers / URIs referring to the same subject.

(Predicate, OntResource, OntResource, OntResource);

For a Predicate occurrence, attributes / values.

(Message, Predicate, OntResource, OntResource);

For a Message Predicate occurrence, possible attributes.

(Context, Message, Predicate, OntResource);

Occurrence (object) for a Context (interpreter) Message (sign) Predicate (concept). Adapter: Context layer (semiotic interpreter).

(Transform, Context, Message, Predicate);

(Mapping, Transform, Context, Message);

(Template, Mapping, Transform, Context);

(Augmentation, Template, Mapping, Transform);

(Resource, Augmentation, Template, Mapping);

(Role, Resource, Augmentation, Template);

(Statement, Role, Resource, Augmentation); Augmentation of which Statement is result of.

(Model, Statement, Role, Resource);

(Entity, Model, Statement, Role); Model (Backends) aligned entities.

(Kind, Entity, Model, Statement);

(Class, Kind, Entity, Model);

(Flow, Class, Kind, Entity);

(Behavior, Flow, Class, Kind);

(Measure, Behavior, Flow, Class);

(Value, Measure, Behavior, Flow);

(Unit, Value, Measure, Behavior);

(Dimension, Unit, Value, Measure);

Example: Application (protocol) shows Dimension, select Unit / Value and assert Measure.

Pick (matched / new) corresponding Behavior. Select available / new Flow.

Pick Flow Class and assign Kind (DCI Role). Assign / create Entity (model alignment / assignation).

Follow up in occurrences hierarchy: CRUD / CUD available / possible. Perform Augmentations.

Adapter (Connector / Client):

Synchronization: Functional. Monads (source / dest: domain / range). Functors (APIs: templates / event drivers for function composition / translation). Inverse functions: backend IO protocols / formats. Adapter endpoint resolution: activates on backends protocols / formats / data.

Adapter (Connector / Client): Model Encoding. Container (reactive message / event driven) APIs. Model / Container APIs interactions.

Adapter / Model "statements" IO abstraction (Forms / Flows: Message events attribute / values).

Message (events): bidirectional CRUD streams (Adapter "template" methods, Model Message declarations):

Adapter: Context layer (semiotic interpreter).

onCreate;

onRetrieve;

onUpdate;

onDelete;