Application:

Workflows (Domain Goals) general purpose ontology matching integration framework.

* Products And Services Community Exchange Network. Resource Oriented Knowledge Computing. Purpose driven Needs / Goods / Products Goals interactions.
* Data, Information, Knowledge exchange: data / schema / behavior Augmentation of virtualized and syndicated / aligned business domains. Business domains applications purpose / problem "spaces" interactions / translations. Addressable interactions: event sourcing.
* Purpose modelling: Business Domains.
* Semantic Hypermedia Browser: declarative front-end / services. Forms / Flows. Annotation, Augmentation & other Domains.

Reference Model:

Encodings:

Reference Model: Key / Value de-referenceable (for matching / embedding purposes) URIs having as host the peer that identified the Resource. DIDs resolution.

* Quads Encoding: (Context, Occurrence, Attribute, Value). Kinds. OGM. Sets / FCA. Context, Metaclass / Role, Class, Instance, Occurrence (Context).
* Matching: Addressable / Browseable Encodings (FCA / Sets) / Identifiers. Order, Dimensional / hierarchical relations (attribute sets). Root Layers (Reference Model) traversal.

Domain Component Model:

Configures runtime (object / functional Domains) for Message parsing and behavior executions.

Domain Declarations: populate layers from Semiotic Context layer Domain description layers resources: Service Resources I/O layers matching / producing semiotic statements for Domain I/O.

Model: Triple Store. Meta Model Schema (RDF / RDFS). Upper Ontology. Primitives.

Controller: Resource Layers object hierarchy API. Named Transforms (Resource URI Service interface / implementation bindings) dataflow: signatures pipelines. Triple Store object graph (DTOs).

View: Layers Domain hierarchy functional API. Messages I/O. Protocol.

Bus. Messages / Layers I/O.

Triple Store:

Meta Model Hierarchy / Schema (Quad Layers / DOM):

CellValue : Value

ColumnField : Sign

ID : occurrence (PK) : Object

Context : instance (table) : Context

Role : metaclass (CSPO)

Resource : class. Monad Value (instance)

Kind : selector / transform (Functor mapping). Monad Value Type (metaclass / role)

Statement (context)

Relation : Kind Grammar (Productions). Monad Instance (occurrence)

Entity : Kind Grammar (Rules). Monad Type (class)

Relationship

Flow

Domain

Resource Layers object hierarchy API:

Functional Layers Domain model / transforms (events / controller). Named Transforms (Resource URI Service interface / implementation bindings) dataflow: signatures pipelines. Triple Store object graph (DTOs).

Value

Sign

Object

Context

Role

Resource

Kind

Statement

Relation

Entity

Relationship

Flow

Domain

* Meta Model: Layers.
* (Context / Relation, Object / Statement, Sign / Kind, Value / Resource).
* Augmentations:
* Aggregation: Rules (Objects).
* Activation: Rules (Signs).
* Alignment: Rules (Values).
* Layers. Roles. Functions (mappings).
* Upper / Lower Layer Roles. Transforms. Bindings (contexts resolution by reference model matchings).
* Intension / Extension. OntResource: Resource reifying aligned data / schema / behavior (data / information / knowledge.
* Type Promotion (roles). Order (dataflow). Reified Relation / Relationship (Production / Rule) context roles / interactions. Matching.
* Layers Resource Context / DOM API. Levels (inheritance hierarchy reification).

TBD.

Layers Domain hierarchy (Functional API):

Monads AST / Parser Builder. Monads: Parsing / Matching, Zippers. Introduction. API: Augmentations, Transforms / Mappings. Traversal. Dataflow.

Layers Monads / Parser Monads (Messages : Rules / Productions). Functional events dataflow (selector signatures : Layer instance Activation).

DOM / AST Hierarchy:

Resource<OntResource[]>;

Kind<Resource[]>;

Statement<Kind[]>;

Relation<Statement[]>;

Entity<Relation[]>;

Relationship<Entity[]>;

Flow<Relationship<Entity[]>;

Domain<Flow[]>;

* Functional Model Monads wraps Meta Model Layers DTOs which represents an endpoint / interface for its Resource URI type and instance.

Bus. Messages / Layers I/O:

Reference Model encoded Messages. Domains streams / pipelines. Messages enter Domains layer through Triple Store and returns back (results, prompts / populated). Messaging backend.

Messages:

(Context, Subject);

(Subject, Predicate);

(Predicate, Object);

Reference Model: Matching (recursive) of aggregated Rules (keys) Productions (values). Domains matches from upper to any level of (inter) Domain activations.

Layers: Productions in one layer are Rules of next layer (context, class, metaclass, instance, occurrence, role, etc.).

Messages flow from Domains layers matching Rules and producing further Productions (Messages) matching lower layers Contexts (Rules) until Service Resource (semiotic) Domain layer (Transform: URI service implementing interface). Transform semantics goes back until Domain layer: Augmentation of Resource layers (prompts / protocol semantics).

* Message: Meta Model Resource Hierarchy (Reference / Semiotic Model) root Layer. Resource Set specification (Rules / Productions). Further Layers implement Message (matching) behavior (polymorphism). Value in role / type / occurrence / context Resource (prompt). Data / schema / behavior I/O.
* Message: Perform Augmentations as with source Domains data (Message Layer Context type, Prompts / Assertions: TBD).
* Aggregation: Match Layers Contexts (Rules Productions).
* Activation: Match Resource Kinds.
* Alignment: Match Resource Values.
* Dataflow: streams, domain / range, endpoints, routes, pipelines. Message I/O (prompts: model / client).
* Bus: Message dispatch. From Aggregation Dataflow matching context subject, kind, resources. Semiotic layer encodes domains graph dataflow.
* Meta Model parsing (by Functional API): Layer Rules (inference prescriptions) / Layers Productions (inference propositions). Parse Message as corresponding Layer Context (for which Message proposition is true for prescriptions).

Augmentations:

* Augmentations Domain:
* Aggregation: clustering stream. JNDI. Registry. Roles in Contexts.
* Activation: classification stream. JAF. Naming. Types in Contexts.
* Alignment: regression stream. Index. Attribute Values in Contexts.
* Model reification: Role Context. Addressable Augmentations (Object extension which is result of Context intension).
* Kinds: Streams of corresponding Roles.
* Semiotic Layer: (Augmentation, Subject, Predicate, Object);
* Aggregation Rules (Layers / Kinds / Resources Mappings):
* (Context, Occurrence, Attribute, Value);
* Map Reduce (Reference Model). Object occurrence of Predicate.
* Reference Model: Map Reduce. Reified Layers. Levels.

Component Domains:

Functional event driven Domains configures models behaviors. Domain types: service resources, interfaces (transforms), signatures (dataflow).

I/O / Persistence Domain.

Sets Augmentations Domain.

FCA Augmentations Domain.

Endpoints I/O Domain.

Predictions Domain.

Dimensional Domain.

Registry Domain.

Index Domain.

Naming Domain.

Business Domains: business specific domain types.

I/O / Persistence Domain:

Events (event sourcing). Backends. Peers. DIDs.

Semiotic Layer: (PersistenceType, PersistenceSubject, PersistenceMember, PersistenceValue);

Type Kind: Domain Service Handler. Domain signatures (domain / range: Subject Kind / Object Kind). Domain graph mappings context handler: function P(S) : O.

Subject Kind: domain persistence resource types (employee).

Member Kind: persistence members resource types (employee/salary;ARS).

Value Kind: range resource types (salary;ARS).

Reify Persistence as Relationship (Values as Relation Resources). Align domain / range with domains / primitive types (Member Kind, salary;ARS).

Event sourcing:

(PersistenceContext, PersistenceContext, PersistenceMember::new, PersistenceSubject);

(PersistenceContext, PersistenceContext, PersistenceMember::delete, PersistenceSubject);

(PersistenceContext, PersistenceSubject, PersistenceMember::delete, PersistenceSubject);

Sets Augmentation Domain:

Semiotic mappings population. Augmentations: Aggregation (layers), Alignment (ontology), Activation (layers dataflows). Render Resource hierarchies.

Semiotic Layer: (SetContext, SetParent, SetSubject, SetChildren);

Context Kind: Domain Service Handler. Domain signatures (domain / range: Subject Kind / Object Kind). Domain graph mappings context handler: function P(S) : O.

Parent Kind: domain resource types (resource).

Subject Kind: attribute resource types (resource/resource).

Children Kind: range resource types (resource).

Reify Sets as Relationship (Values as Relation Resources).

Sets API: Augmentations, Transforms / Mappings. Traversal (Context, Statement, Kind, Resource).

FCA Augmentations Domain:

Semiotic mappings population. Augmentations: Aggregation (layers), Alignment (ontology), Activation (layers dataflows). Objects / attributes objects / concepts traversal. Render Resource hierarchies.

Semiotic Layer: (FCAContext, FCASubject, FCAAttribute, FCAValue);

Context Kind: Domain Service Handler. Domain signatures (domain / range: Subject Kind / Object Kind). Domain graph mappings context handler: function P(S) : O.

Subject Kind: domain resource types (resource).

Attribute Kind: attribute resource types (resource).

Value Kind: range resource types (resource).

Reify FCA Context as Relationship (Values as Relation Resources).

* FCA API: Augmentations, Transforms / Mappings. Traversal (Concepts, Objects, Attributes, Products).
* FCA Contexts:
* Reference Model.
* Meta Model.
* Encodings.
* Domain Models Augmentations.

Endpoints Domain:

Streaming I/O Dataflow.

Semiotic Layer: (EndpointContext, EndpointSubject, EndpointRequest, EndpointResponse);

Context Kind: Domain Service Handler. Domain signatures (domain / range: Subject Kind / Object Kind). Domain graph mappings context handler: function P(S) : O.

Subject Kind: domain resource types / referrer (employment).

Request Kind: request resource types (person).

Response Kind: response range resource types (employee).

Reify Endpoint as Relationship (Values as Relation Resources).

Augmented Resources Contexts / Interactions Services.

Forms / Flows (Grammar / Protocol Builder. Prompts). Resource augmentation endpoints. Forms / Flows browsing APIs. DCI: Declarative Forms / Flows.

OGM / Client Drivers Services.

* REST: Current / referrer. Rel. HREF. Link body. Metadata. Endpoint Domain.

Predictions Domain:

Semiotic Layer: (PredictionType, PredictionSubject, PredictionItem, PredictionValue);

Type Kind: Domain Service Handler. Domain signatures (domain / range: Subject Kind / Object Kind). Domain graph mappings context handler: function P(S) : O.

Subject Kind: domain resource types (image).

Item Kind: prediction resource types (image/face).

Value Kind: range resource types (face).

Reify Prediction as Relationship (Values as Relation Resources).

Dimensional Domain:

Semiotic Layer: (DimensionType, DimensionSubject, DimensionItem, DimensionValue);

Type Kind: Domain Service Handler. Domain signatures (domain / range: Subject Kind / Object Kind). Domain graph mappings context handler: function P(S) : O. Time example (contains / before).

Subject Kind: domain resource types (hour; dayOfWeek) : 1. Monday.

Item Kind: dimensional resource types (hour/minutes; dayOfWeek/dayOfWeek) relations: contains / before.

Value Kind: range resource types (minutes; dayOfWeek) : 60. Tuesday.

Reify Dimension as Relationships (Values as Relation Resources).

Upper Ontology: relations / primitives.

* Dimensional: inference / relation types / restrictions.
* (Context, Occurrence, Attribute, Value);
* (Context, Class, Metaclass, Occurrence); Instance, Role.
* Encode order / hierarchies / relations (parent / child, prev / next, etc.) / iterations / conditionals / jumps.
* Dimensional Domain: units, measures, values. Comparisons, relations. State. Events (marriage example). Verbs (action, passion, state). Order (data / schema / behavior).

Registry Domain:

Key / Value for graph contexts, nodes, predicates.

Semiotic Layer: (RegistryType, RegistrySubject, RegistryKey, RegistryValue);

Type Kind: Domain Service Handler. Domain signatures (domain / range: Subject Kind / Object Kind). Domain graph mappings context handler: function P(S) : O.

Subject Kind: domain node resource types (person).

Key Kind: registry resource types (person/age;int).

Value Kind: range value resource types (age).

Reify Registry as Relationships (Values as Relation Resources). Align domain / range with primitive types (Key Kind, age;int).

Index Domain:

Indexing of graph contexts, nodes, predicates.

Semiotic Layer: (IndexType, IndexTerm, IndexScope, IndexValue);

Type Kind: Domain Service Handler. Domain signatures (domain / range: Subject Kind / Object Kind). Domain graph mappings context handler: function P(S) : O.

Term Kind: domain node resource types (resource).

Scope Kind: dimensional resource types (resource/resource).

Value Kind: range value resource types (resource).

Reify Index as Relationships (Values as Relation Resources).

Naming Domain:

Terms translation in contexts for graph contexts, nodes, predicates. Alignment / matching.

Semiotic Layer: (NamingType, NamingSubject, NamingContext, NamingValue);

Type Kind: Domain Service Handler. Domain signatures (domain / range: Subject Kind / Object Kind). Domain graph mappings context handler: function P(S) : O.

Subject Kind: domain node resource types (term).

Context Kind: dimensional resource types (term/term).

Value Kind: range value resource types (term).

Reify Naming as Relationships (Values as Relation Resources).

Domains Dataflow:

Layers Dataflow: Augmentation. Rules / Productions matching (Reference Model / Kinds Aggregation).

Semiotic Dataflow: Object Kind matches Subject Kind of Context Kind signatures. (Sucessive Layers Dataflow).

Ontology Matching:

Upper Ontology. Grammars. Primitives.

Matching: Resource occurs as context / occurrence / atribute / value or class / occurrence / context / metaclass / instance in equivalent occurrence contexts (kinds / order / shapes / type hierarchies).

Meta Model encodes mappings for equivalence / relations hierarchies for entities instance occurrences in roles in contexts for concepts recursively till upper onto / primitive terms / relations.

Reify relation from / to predicates (semiotic) / relation entity (expanded relation entity / roles statements). Dimensional measures / state events. Shapes: transforms / rules.

FCA Ontology Matching: Upper ontology / primitives. Reference Model objects / attributes encoding. Encoding (scaling): lattice concepts relations / transforms traversal.

Semiotic mappings population. Augmentations: Aggregation (layers), Alignment (ontology), Activation (layers dataflow transforms: context products).

Reference Model Contexts.

Meta Model Layers Contexts.

* Alignments (Reference Model types / values):
* Data Alignment: key / val.
* Schema / Information Alignment: tuples.
* Behavior / Knowledge Alignment: dimensional.