**1: Mision / Vision**

**1.1: Objectives**

**1.2: Description**

**2: Use Cases**

**2.1: Problems**

**2.3: Solution**

**3. Approach**

**3.1: Services / Features**

**3.1.1: Reactive / Event Driven**

Message based Augmentation Events Dataflow. Augmentation Mapping Dataflow allowing to embed dynamic state in Model entities (including Mappings Augmentations themselves).

**3.1.2: Augmentation Dataflow**

Functional declarative way of stating Augmentation Transforms over Messages / Resources matching / populated by input Templates performing output Mappings Augmentation reflecting input, model and behavior state.

**3.1.3: Ontology Matching**

Determine whether two identifiers refer to the same entity, whether two relations are the same and which results corresponds to instances of the same actions.

**4: RDF Introduction**

**5: RDF Quads / Objects Mapping**

As RDF Quads encodes four URI values (CSPO Statement) an Object - RDF Quad elemental mapping could be implemented regarding an RDF Quad Statement CSPO as follows:

(C: Context, S: Occurrence, P: Attribute, O: Value);

where Context (C) is the URI of an Object Class identifier, Occurrence (S) is the URI of an Object Class Instance identifier and, aggregating same Class / Instance pairs, Attribute (P) and Value (O) are, respectively, Class Instance member (name, domain / range) and values for the aggregated (S) Object of Class (C).

Contexts. Occurrences, Attributes, Values: Roles of Meta Resource(s) in contexts.

Subject in Statement has Predicate and Object Attribute / Value (roles).

Predicate in Statement has Subject and Object Attribute / Value (roles).

Object in Statement has Subject and Predicate Attribute / Value (roles).

Value as Occurrence of Attribute in Attribute Occurrence Context.

Context Kind (signature): Subject Kind and Object Kind Attribute / Value (roles).

Subject / Occurrence / Context / Role : Attribute, Value. Concepts. Semiotic Metamodel. Dimensional Encoding: each type as each (pair) kind. Pairs (tags / facets).

Meta Model: Layers Resource relations:

Instance, class, metaclass, occurrence, role. DOM, Actor / Context / Role.

Layer Context: Statement class. Aggregates same Context Statement(s). Next layer metaclass (Occurrence)..

Layer Occurrence: Statement Context metaclass. Aggregates same Context / Occurrence Statement(s). Previous layer context.

Layer Attribute: Statement Context Ocurrence Attribute (occurrence). Previous layer Occurrence.

Layer Value: Statement Context Occurrence Attribute Value (role). Previous layer Attribute.

Layer Aggregation begins with Model initial Statement having a new Context (class) “pushing” previous CSPO right, being the new class the new layer Context and CSP becoming SPO:

(C, S, P, O)

(N, C, S, P).

Functional / Object Oriented Resource API (Model, Statement, Semiotic, Dimensional layers, Meta Resources).

**6: Models**

Source / Grammar / Pragma Levels.

Functional / Dimensional / Semantic Facets.

Reactive Entities: Resource, Model, Message, Kind.

Entities: ID (routes), State (ctx / rel pointers, occurrences). Streams, Dataflow (routes / bindings: addressing).

Transforms, Augmentation (functors / mappings). Dataflow: Message / Model / Augmentation / Model / Message.

Meta Model (Interaction Layer Augmentation Aggregated Model declarations: facets, levels, layer).

Meta Model Interaction Layer (Augmentation: aggregated Source, Grammar, Pragma Levels Mappings) Mappings render Data, Schema, Behavior Resources for Functional, Dimensional, Semantic Meta Model Facets layers.

Model Layers:

Data Layer (aggregate Message in Interaction Mapping Augmentation context).

Schema Layer (aggregate Data).

Behavior Layer (aggregate Schema).

Model Levels:

Source Level (Functional Facet).

Grammar Level (Dimensional Facet).

Pragma Level (Semiotic Facet).

Model Facets:

Functional Facet.

Semiotic Facet.

Dimensional Facet.

Models aggregates Message input IO / Connectors data into corresponding knowledge Facets (Functional, Semiotic, Dimensional). Model is a layered structure of RDF Quads which follow the base (Functional) Model structure:

OntResource is the class responsible for aggregating different URIs referring the same entities (Ontology Matching).

Resource : Functional (Monad) OntResource wrapper.

(Context : Resource, Occurrence : Resource, Attribute : Resource, Value : Resource);

(Statement, Occurrence, Attribute, Value);

(Entity, Statement, Occurrence, Attribute);
(Role, Entity, Statement, Occurrence);
(Class, Role, Entity, Statement);
(Flow, Class, Role, Entity);
(Behavior, Flow, Class, Role);

Models have layer statements in which statement context (Facet Roles) classes are a hierarchy from Resource to Behavior and where context role instances follow a hierarchy of a dynamic type system (Kinds).

Input Layer (Resource).

Data Layer (Statement instance, Entity class).

Schema Layer (Kind / Role instance, Class class).

Behavior Layer (Flow instance, Behavior class).

The idea is to infer Schema (classes and instance of classes / relations) operating over Data layer. Then, by aggregating Data and Schema, infer Behavior (classes and instances of operations / functions). This Aggregation, together with Alignment and type Activation mechanisms comprehend the Model core Augmentations.

Context / Resource type hierarchy design pattern: plain class hierarchy, parameterized class on Resource(s) / URIs, monads, metaclass, others. Actor / context / role (Statement CSPO position / Meta Resource). Reified Model types. DOM.

Meta Resource(s): URI, Resource, Statement, CSPO, Context / Layer, Occurrence, Attribute, Value, Kind, etc.

DOM, Actor / Role / Context, OGM APIs.

Augmentation: transform algorithm (basic operation).

Encoding: Model (Resource).

Model: RDF Backend.

URIs Services: API for plugging whatever connector may be implemented for behaving in a reactive message oriented fashion (back ends).

Resource: Abstracts (wraps) URIs Services in a functional API (Resource streams). DOM, Actor / Context / Role (Meta Resources).

Augmentation: Parse Message (event: context quad) according Template (pattern), materialize output Transform. Algorithm (TBD): case classes, pattern matching, destructuring, Resource monad chained operations (Template: functor) functional streams, ADTs.

Dataflow, Reactive: Resource Monad handling of wrapped URIs messages / events I/O via HTTP verbs. Augmentation: Model, Context instance / class (layers), Resources producing / reacting to events. Endpoints: Discovery / Location / Resolution services. URI APIs (signatures discovery).

Meta Graph / Model, Meta Resource(s): Resources / Messages reifying "patterns" on inputs (URI, Resource, Statement, Kind(s), Context, Occurrence, Attribute, Value, Layer Context classes, etc.). Declarative statement for Augmentation shapes applyied to input contexts.

Meta Model default Augmentations:

Aggregation classification. Registry svc.

Alignment regression. Index svc.

Activation clustering. Naming svc.

Context Kind Signatures.

Datasources / Backends / Services. URIs. Signatures: dataflow (Context Kinds). CKs Attribute / Value (SK / PK) determines domain / range I/O of a Resource / URIs.

Ontology matching (Backend / Interaction Model).

Model Meta Resource: Model components reified Resource types / instances (URIs, Resource, Statement, Context : Layer, Kind, etc.). Augmentation templates "placeholders" (signatures, matching of common upper resources).

Kinds (Application):

Kind: Basic type inference. Applied over layers CSPO during Activation Augmentation. An Occurrence Attributes / Values, aggregated for its URI and Context, determines Kind "members" (Attribute) and Kind instance member values (Value).

Super Kind / sub Kind hierarchy relationship is given by a set of Kind Attributes being super set / sub set of each other.

Examples.

SubjectKind (meta Resource): For a given URI occurring as Subject (Occurrence) across a set of Statements (Contexts), its aggregated Predicates (Attributes) defines its "Kind" and its Attribute values determines the given Kind instance "members" values.

ObjectKind (meta Resource): for a given URI occurring as Object (Value) over a set of Statements, Subject (Kind Attribute), Predicate (Kind Value).

PredicateKind (meta Resource): for a given URI occurring as Predicate over a set of Statements, Object (Kind Attribute), Subject (Kind Object).

ContextKind: SubjectKind (Attribute), ObjectKind (Value). Context (Statement) "signature" (dataflow inputs / outputs activation: domain / range).

Models aggregates input I/O / Connectors data into corresponding knowledge Facets (Functional, Semiotic, Dimensional).

Base Model structure / Context layers hierarchies is as follow:

OntResource (URIs).

Resource : Functional URI wrapper.

(Context : Resource, Occurrence : Resource, Attribute : Resource, Value : Resource);

(Statement, Occurrence, Attribute, Value);

(Entity, Statement, Occurrence, Attribute);
(Role, Entity, Statement, Occurrence);
(Class, Role, Entity, Statement);
(Flow, Class, Role, Entity);
(Behavior, Flow, Class, Role);

Models have layers in class / instance roles (except for input layer) and each upper layer aggregates functionally over the previous:

Model (Facet) Statement declaring /aggregating Model in Meta Model is of the shape:

(Model : Model Impl., Behavior, Flow, Class); Interaction / Meta Model.

Classifying (aggregating) previous layers statements as parts of the Model.

Input Layer (Resource).

Data Layer (Statement instance, Entity class).

Schema Layer (Kind / Role instance, Class class).

Behavior Layer (Flow instance, Behavior class).

The idea is to infer Schema (classes and instance of classes / relations) operating over Data layer. Then, by aggregating Data and Schema, infer Behavior (classes and instances of operations / functions). This Aggregation, together with Alignment and type Activation mechanisms comprehend the Model core Augmentations.

Models:

URI(s);

OntResource; Merged URI(s) wrapper.

Resource (OntResource CSPO / Contexts hierarchies Monad wrapper);

Message (Resource Monad wrapper); Request / Response Encoding.

Template / Transform (Message blueprints) domain / range : Message.

Augmentation : Functor.

Mappings: Declarative IO signatures: Context Kinds Templates / Transforms. Subscriptions / routes. Dataflow.

Encoding: Template Message augmentation (inputs).

Encoding: Declarative functors behavior encoding statements. Mappings (subscription / routes).

Encoding: Transform Message augmentation (outputs).

Interaction Model:

(Augmentation, Template, Mapping, Transform);

Dataflow: Order, Flows (Mappings, hierarchies).

**6.1: Model Layers**

What my attempts where about in the beginning was to match different URIs or, for example, database identifiers which refer to the same entity (in different databases / ontologies, for example) to perform some kind of "ontology matching".

Then I've tried to develop a mechanism for using RDF Quads for encoding an object graph (and a layers class hierarchy) using Contexts to denote the class of an instance, Subjects to denote class instances and attributes (members) and values: Predicates / Objects.

Then I've realized that some basic type inference could be performed with, for example, aggregating Subjects with the same predicates (Subject Kinds). Idem for Predicates, Objects and Contexts. I've also realized that plain "facts" statements could be aggregated in the previously mentioned class hierarchy to abstract further, from plain data, instance / class layers of what I call data / schema / behavior layers. Higher layers (i.e.: Behavior) "aggregate" lower layers.

Quads are "reified" as Resource(s). Also, Resource is a functional wrapper reactive and event driven of an URI. And an URI could be implemented with whatever backend which could produce or consume events (databases, services, etc.). Resource layers hierarchy (Context) is to be implemented by an actor / role type object pattern according the hierarchy layer level it corresponds (and declaratively stated in a Model of Meta Resources).

Layers shape is as follow:
Resource : Functional URI wrapper.

(Context : Resource, Occurrence : Resource, Attribute : Resource, Value : Resource);

CSPO Names are according roles (Meta Resource) in the Model. For example: layer Occurrence is parent layer class.

Each layer abstract instances of its own contexts instances.

Input Layer: (CSPO layer):

(Transaction, someOne, buys, someProduct);

Statement (data layer instance):

Inputs regarding the same context are aggregated into data layer instance.

(Statement, Occurrence, Attribute, Value);
(transactionStatement, someOne, buys someProduct);

Entity (data layer class):

Aggregated Statement and Occurrence Statement occurrences reified into an Entity along with its Occurrences Attributes.

(Entity, Statement, Occurrence, Attribute);

(someTransaction, transactionStatement, someOne, buys);

Role / Kind (schema layer instance):

Aggregated Entity and Statement Entity occurrences reified into a Role / Kind along with its Statements and Occurrences.

(Role / Kind, Entity, Statement, Occurrence);
(someBuyer, someTransaction, transactionStatement, someOne);

Class (schema layer class):

Aggregated Role and Entity Role occurrences reified into a Class along with its Entities and Statements.

(Class, Role, Entity, Statement);
(Person, someBuyer, someTransaction, transactionStatement);

Flow (behavior layer instance):

Aggregated Class and Role Class occurrences reified into a Flow along with its Roles and Entities.

(Flow, Class, Role, Entity);
(someBuy, Person, someBuyer, someTransaction);

Behavior (behavior layer class):

Aggregated Class and Role Class occurrences reified into a Behavior along with its Classes and Roles.

(Behavior, Flow, Class, Role);
(Buy, someBuy, Person, someBuyer);

Then, each Model aggregates its Statements in the form (for example):

(Model Impl, Buy, someBuy, Person); Interaction / Meta Model.

This "aggregations" are part of what I call "Augmentation(s)": Aggregation, Alignment and Activation are ones of those, which are functional transforms described declaratively in an object graph Meta Model. The act of applying an Augmentation implies one source Message Resource (context layer), one matching Template Resource (input signature) an Augmentation (Interaction functor) a Transform Resource (output signature) and a resulting (set of) Message Resource(s) materialized as further layers instances / Messages to be “parsed” by further corresponding Augmentations of matching Template signatures (dataflow).

One also could Augment Resource(s) in a functional manner, using reactive event driven APIs so, for example applying "Person" class to "Employee" role could shield a Resource set of people being working for someone. The ultimate goal is to be able to "plug" as much "backends" connectors as posible into distributed peers which exposes protocols / APIs for knowledge driven hypermedia applications.

**6.1.1: Interaction Layer**

(Augmentation, Template, Mapping, Transform);

Model declared as Interaction Model Augmentation (matching Messages) in Interaction Model. Flows:

Message

Model (Functor)

Augmentation (Addressable Interaction)

Template (Message)

Mapping (Functor)

Transform (Message)

Model (Functor)

Message

Interaction Model: Model Events (Augmentation).

Augmentation: Event. Signature. Declarations / Occurrences. Domain, Input / Mapping, Transform / Range, Output.

Embeddings: Message Match Event Signature. Tempate matching / Transform rendering. Dataflow.

Embeddings: Meta Model Augmentation, Template, Mapping, Transform Meta Resources (input layer). Meta Model Source, Session, Interaction levels.

Model OntResource Augmented with Event Transform aggregates new Event Mapping.

Encoding. Addressing (contents, signatures, contexts). Events publish / subscribe. Dynamic subscriptions / bindings. Subscription, reactive Meta Resource(s). Message flow mechanism: from Model to base layers.

Match Message subjects to Templates, Augmentations and Transforms roles (bound by CK signatures dataflow). Message inputs: Models.

Interaction Model for Encoding / Addressing (Mapping : Event routes) Dataflow metadata.

Augmentation: Described in Interaction Model. Encode Message, Template, Augmentation and Transforms roles (Meta Resources). Data, Session, Interaction Levels (Message, Template, Transform, Augmentation statements declaration realization).

Augmentation. Aggregation Meta Model: Describe layers contexts compositions. Alignment Meta Model: Describe augmented attributes (by kinds clustering). Activation Metamodel: Describe Kinds / Roles activation (by attributes aggregations).

(Augmentation, Template, Mapping, Transform);

Model declared as Interaction Model Augmentation (matching Messages) in Interaction Model. Flows:

Message

Model (Functor)

Augmentation (Addressable Interaction)

Template (Message)

Mapping (Functor)

Transform (Message)

Model (Functor)

Message

Interaction Model: Model Events (Augmentation).

Augmentation: Event. Signature. Declarations / Occurrences. Domain, Input / Mapping, Transform / Range, Output.

Embeddings: Message Match Event Signature. Tempate matching / Transform rendering. Dataflow.

Embeddings: Meta Model Augmentation, Template, Mapping, Transform Meta Resources (input layer). Meta Model Source, Session, Interaction levels.

Model OntResource Augmented with Event Transform aggregates new Event Mapping.

**6.1.1: Data Layer**

**6.1.2: Schema Layer**

**6.1.3: Behavior Layer**

**6.2: Model Facets**

Models have “Facets” which renders the different ways Model data / schema / behavior could be regarded and used for different purposes, from application development to Business Intelligence and Ontology Matching.

Facets are models implemented the same way other models are with Model Resource Contexts and layers and from the same data. Each Facet implements its own Resource URI wrapper (same URIs, ontology matching, provenance of aligned URIs, Facet pivoting). Then, each Facet has its own Model Context Resource class hierarchy having Augmentation / Dataflow functors as Model Resource(s) does.

**6.2.1: Functional Facet**

(Context : Resource, Occurrence : Resource, Attribute : Resource, Value : Resource);

(Statement, Occurrence, Attribute, Value);

(Entity, Statement, Occurrence, Attribute);
(Role, Entity, Statement, Occurrence);
(Class, Role, Entity, Statement);
(Flow, Class, Role, Entity);
(Behavior, Flow, Class, Role);

(Model, Behavior, Flow, Class);

**6.2.2: Semiotic / Semantic Facet**

(Context : Resource, Occurrence : Resource, Attribute : Resource, Value : Resource);

(Attributes, Occurrence, Attribute, Value);

(Object, Attributes, Occurrence, Attribute);
(Concept, Object, Attributes, Occurrence);
(Sign, Concept, Object, Aytributes);
(Context, Sign, Concept, Object);
(Interaction, Context, Sign, Concept);

(Model, Interaction, Context, Sign);

**6.2.3: Dimensional Facet**

(Context : Resource, Occurrence : Resource, Attribute : Resource, Value : Resource);

(Properties, Occurrence, Attribute, Value); Data (Properties: distance / facts).

(Value, Properties, Occurrence, Attribute); Info (Properties distance between Occurrence / previous and Occurrence / next).
(Measure, Value, Properies, Occurrence); Knowledge.
(Unit, Measure, Value, Properties);
(Dimension, Unit, Measure, Value);
(Concept, Dimension, Unit, Measure);

(Model, Concept, Dimension, Unit);

Example:

(Value, Previous, Distance, Next); Person, Single, Marriage, Married; Man, Single, Marriage, Husband; Woman, Single, Marriage, Wife.

Order layers statements. Hierarchies (contexts / kinds). Parent / child relationships (steps). Order type relationships: husband: single / marriage / married.

(Value, Previous, Distance, Next); Person, Single, Marriage, Married; Man, Single, Marriage, Husband; Woman, Single, Marriage, Wife.
(Measure, Value, Previous, Distance);
(Unit, Measure, Value, Previous);
(Dimension, Unit, Measure, Value);
(Concept, Dimension, Unit, Measure);
(Resource, Concept, Dimension, Unit);
(Statement, Resource, Concept, Dimension);

Value, Previous, Distance, Next. Dimension, Unit, Measure, Value (aggregated ordered statements layers).

Value -> distance(prev, next); ordering;

Assert knowledge: 1h -> 60min;

dom-lun-mar-mie-jue-vie-sab (orders);

1mt -> 100cm;

etc.

Comparison / order: Alignments (prev, curr, next asserted knowledge). Next hour, location, city, country, next distance at next time at current speed. Event sourcing / tracking: married -> marriage occurred.

Sort: cause / effect, temporal, etc. Messages align, functional map, fold, etc. Primitives. Encode layered statements ordering. Complement / supplement concepts definitions.

**6.3: Model Ontology Levels**

Models have “Ontology” Levels. Levels are Layers (of the Model) which are feed into its input Layer with (instance) Statements aggregated from initial input data (Data Level) aggregated into subsequent layers. Schema Level instances feeds the Model input conforming a Session (context / grammars) ontology Level. Then, behavior Level instances feeds the Model input conforming an Interaction (behavior) ontology Level.

**6.3.1: Model Source Level (Backend)**

Input Statements coming from plain RDF Quads aggregated according Data / Schema / Layers Augmentation(s). Base facts Model Level.

**6.3.2: Model Session Level**

Aggregate Source (Backend) Level Schema layer Statements as Model Session level Data layer input. Reify Schema (roles / grammars).

**6.3.3: Model Interaction Level**

Aggregate Session Level Behavior layer Statements as Model Data level Data layer input. Reify behaviors (context / interactions).

Declarative application protocol use case upper ontology levels (Action… Gesture, etc).

**6.4: Model Monad**

**6.5: Functional APIs**

**6.6: Streams (Contexts, Kinds, filters)**

**6.7: Transforms (Meta Model).**

**7: Meta Model**

URI;

Resource (URI\*);

Role (Model CSPO hierarchies) : Resource;

Statement (Resource, Resource, Resource, Resource) : Resource;

Kind (Statement\*) : Resource;

Class (Kind\*) : Resource;

Context (Class\*) : Resource;

Hierarchy: class (Object / Value) as superclass Context.

Object: class (extension);

Context: super class (intention);

(Kind, Statement, Role, Resource); Data (Resource Kind).

(Class, Kind, Statement, Role); Schema (Role Class)

(Context, Class, Kind, Statement); Interaction (Statement Context).

State Facet / Layer / Level / Augmentation / Model Resource Mappings.

Functional API: Message IO. Mappings.

State order (in context class hierarchies axes), comparison relations, iterations, flow, events, causal relations, units, enums, equivalence, etc.

Data order: Resource Kind hierarchies.

Schema order: Role Class hierarchies.

Interaction order: Statement Context hierarchies.

URI(s);

OntResource; Merged URI(s) wrapper.

OntResource hierarchy: layers statement contexts. Facets DOM, Actor / Role.

Resource (OntResource Context Roles hierarchies Monad wrapper);

Resource (OntResource CSPO / Contexts hierarchies Monad wrapper);

Role (Model CSPO Context Roles hierarchies type classes) : Resource;

Statement (Resource, Resource, Resource, Resource) : Resource;

Kind (Statement\*) : Resource;

Class (Kind\*) : Resource;

Context (Class\*) : Resource;

Hierarchy:

Object: class (extension);

Context: super class (intention);

(ID, ID, ID, ID);

(Transform, ID, ID, ID);

(Mapping, Transform, ID, ID);

(Template, Mapping, Transform, ID);

(Augmentation, Template, Mapping, Transform);

(Model, Augmentation, Template, Mapping);

(Resource, Model, Augmentation, Template);

(Role, Resource, Model, Augmentation);

(Statement, Role, Resource, Model);

(Kind, Statement, Role, Resource);

(Class, Kind, Statement, Role);

(Context, Class, Kind, Statement);

State Facet / Layer / Level / Augmentation / Model Resource Mappings.

Meta Resources layers contexts class hierarchy:

Context : Class : Kind : Statement : Role : Resource : Model : Augmentation : Template : Mapping : Transform : ID;

Meta Resources layers Reification: Context : Class : Kind : Statement : Role : Resource : Model : Augmentation : Template : Mapping : Transform as ID Context Statement (ID statements for each context layer). Model Levels (Facets / Levels). Aggregate reifications.

Functional API: Message IO. Mappings.

Interaction Model (Interaction Level):

(Augmentation, Template, Mapping, Transform);

Dataflow: Order, Flows (Mappings, hierarchies).

State order (in context class hierarchies axes), comparison relations, iterations, flow, events, causal relations, units, enums, equivalence, etc.

Data order: Resource Kind hierarchies.

Schema order: Role Class hierarchies.

Interaction order: Statement Context hierarchies.

Meta Model: encode Layers, Contexts, Kind / Roles hierarchies (subject, context, occurrence, roles, atributes, values / metaclass, class, instance relations / meta resources) and Facets using corresponding Facets implementations of base Model Meta Resources.

Functional API: Message IO. Mappings.

Model state: Context (Resource : data), Kind (Grammar : schema), Dimension (behavior). Context Kind(s) signatures: Dataflow.

Augmentation: basic operation.

Monad: Resource<URI>.

Resource layers hierarchy API.

Data / Reference Model. Model Functional Semantics (Model / Layer / Message application). Augmentation: Basic Model I/O operation. Message spec / Resource Set Specification (result).

Service URIs:

Service URIs: Context Kind (inputs / outputs domain / range). Example: predictions, classification, clustering, regression. Index / Naming / Registry "contexts" (facets).

Extended content types activations on domain / range (verbs, augmentations). Example: image, face, crop.

Functional Resource Model / Context / Attributes / Kind design / implementation. Serialization (Encoding / Models). Signatures. Reactive. Augmentation. DOM, Actor / Context / Role. APIs: Augmentation. Meta Resources.

Meta Model: Encode / reify Model(s) declaratively w./ Meta Resources and Model Context(s) hierarchies.

Meta Model: Encode Kind / Context hierarchies.

Meta Model: Encode order, iteration, conditional flow. Dataflow.

Functional Resource Model / Context / Attributes / Kind design / implementation. Serialization (Encoding / Models). Signatures. Reactive. Augmentation. DOM, Actor / Context / Role. APIs: Augmentation.

Resources API hierarchy.

Meta Resources.

Meta Model: Encode / reify Model(s) w./ Meta Resources and Model Context(s) hierarchies.

Meta Model: Encode Kind / Context hierarchies.

Meta Model: Encode order, iteration, conditional flow. Dataflow.

Augmentation / Models: Source, Grammar, Dimensional Models. Core Meta Model Augmentation Template(s): Encoding signatures Dataflow.

Functional Resource Model / Context / Attributes / Kind design / implementation. Serialization (Encoding / Models). Signatures. Reactive. Augmentation. DOM, Actor / Context / Role.

Meta Resources.

Meta Model: Encode / reify Model(s) w./ Meta Resources and Model Context(s) hierarchies.

Meta Model: Encode Context hierarchies.

Meta Model: Encode order, iteration, conditional flow. Dataflow.

Encoding: Kind hierarchies / Grammars (CK, SK, PK, OK).

Encoding / Models: Source, Dimensional Models. Encoded Grammar Template(s).

Augmentation: declaration (signatures) / algorithm.

Ontology Matching. Semiotic. Sets. Functional Reference Model.

Meta Model: encode Layers, Contexts, Kind / Roles hierarchies (subject, context, occurrence, roles, atributes, values / metaclass, class, instance relations / meta resources).

Augmentation: Described in Meta Model. Encode Message, Template, Augmentation and Transforms roles (Meta Resources). Data, Session, Interaction Levels (Message, Template, Transform, Augmentation statements declaration realization).

Augmentation. Aggregation Meta Model: Describe layers contexts compositions. Alignment Meta Model: Describe augmented attributes (by kinds clustering). Activation Metamodel: Describe Kinds / Roles activation (by attributes aggregations).

Encoding. Addressing (contents, signatures, contexts). Events publish / subscribe. Dynamic subscriptions / bindings. Subscription, reactive Meta Resource(s). Message flow mechanism: from Model to base layers.

Match Message subjects to Templates, Augmentations and Transforms roles (bound by CK signatures dataflow). Message inputs: Models.

Reify Model Layers, Levels and Facets in a Meta Model with Meta Resources. Use Meta Resources class relations for describing models. Meta Resources describe components and roles of Models according a set of relations:

Subject (Resource) / Context (Statement) / Occurrence (CSPO instance) / Role (Kind) / Attribute / Value.

Metaclass (Occurrence) / Class (Context) / Instance (Attributes / Values).

The aim is being able to describe models using models themselves, maybe translating relations to Model Quad Statements.

The same relations could be used to build a Model in which declaratively state model dataflow behavior (reaction to events). A dataflow specification could be described by the following meta resources (roles):

Message (Subject : Data level)

Template (Context / domain : Session level)

Augmentation (Occurrence, declarative / service Resources: functors. Interaction level)

Transform (Role / range: Kind transform matches. Session level). Resulting Message Attribute / Value roles populated.

Meta Model:

Meta Resource class / instance patterns.

Participation: Subject in Occurrence.

Role: Participation for Subject.

Kind / Context hierarchies.

Subject, Participation, Occurrence, Roles, Atributes, Values / Metaclass, Class, Instance class / relations / meta resources.

(Participation, Role, Attribute, Value);

(Subject, Participation, Role, Attribute);

(Occurrence, Subject, Participation, Role);

Mappings: Facets (Models / Contexts declarations) by Meta Resource statements in Meta Model. Mappings renders Model(s) contents statements (layers) by Context Augmentations.

Augmentations defined as declarative Mappings in Meta Model encoding Context (layer) inputs matching signatures and augments current / previous layer emmiting mapping transforms. Context : Functor. Participation wraps Context / Resource.

Context::flatMap(ctx : Context) : Context

Aggregation (Augmentation): Apply each Context (layer) Functor on inputs (from input layer) and emits Transform, matching corresponding (next) layer. Next layer Context and SPO according functional mapping declared by Meta Resource types on augmented layer.

Alignment (Augmentation): ToDo.

Activation (Augmentation): ToDo.

Meta Model for Encoding / Addressing (Event routes) dataflow metadata.

Meta Model:

URI;

Resource (URI\*);

Role (Model CSPO hierarchies) : Resource;

Statement (Resource, Resource, Resource, Resource) : Resource;

Kind (Statement\*) : Resource;

Class (Kind\*) : Resource;

Context (Class\*) : Resource;

Hierarchy: class (Object / Value) as superclass Context.

Object: class (extension);

Context: super class (intention);

(Kind, Statement, Role, Resource); Data (Resource Kind).

(Class, Kind, Statement, Role); Schema (Role Class)

(Context, Class, Kind, Statement); Interaction (Statement Context).

State Facet / Layer / Level / Augmentation / Model Resource Mappings.

Functional API: Message IO. Mappings.

State order (in context class hierarchies axes), comparison relations, iterations, flow, events, causal relations, units, enums, equivalence, etc.

Data order: Resource Kind hierarchies.

Schema order: Role Class hierarchies.

Interaction order: Statement Context hierarchies.

Models:

URI(s);

OntResource; Merged URI(s) wrapper.

Resource (OntResource CSPO / Contexts hierarchies Monad wrapper);

Message (Resource Monad wrapper); Request / Response Encoding.

Template / Transform (Message blueprints) domain / range : Message.

Augmentation : Functor.

Mappings: Declarative IO signatures: Context Kinds Templates / Transforms. Subscriptions / routes. Dataflow.

Encoding: Template Message augmentation (inputs).

Encoding: Declarative functors behavior encoding statements. Mappings (subscription / routes).

Encoding: Transform Message augmentation (outputs).

(Augmentation, Template, Mapping, Transform);

Dataflow: Order, Flows (Mappings, hierarchies).

**7.1: Meta Resources**

Meta Resource / Meta Model:

Meta Resource / Model: encode Model, URIs / Layers / Contexts / Facets / Levels / Resources hierarchies. Mappings.

Meta Resource / Model: Encode Message, Template, Augmentation(s), Transforms and Mappings (Dataflow).

Meta Model: Data, Session, Interaction Levels (Message, Template, Transform, Augmentation statements). Mappings.

Model Context / Layers, Facets, Ontology levels, Meta Resources / Models mappings / reification. APIs. Levels example: Behavior / Interaction (Action, Gesture..., Flow). Upper ontologies: Action, Gesture etc. classes.

Contexts / Layers / Levels / Facets Meta Resources / Models classes / instances hiers (ontology matching / data, schema, behavior alignments). Members: URIs, Resource, Context, CSPO, Meta Resource / Model APIs.

Meta Resources are used by a Model Meta Model for describing models. Some of them are:

URI

Resource

Context / Context

Subject / Occurrence

Predicate / Attribute

Object / Value

Statement

Model

Kind

ContextKind

SubjectKind

PredicateKind

ObjectKind

Message

Template

Augmentation

Transform

Class

Metaclass

Instance

Meta Model:

URI;

Resource (URI\*);

Role (Model CSPO hierarchies) : Resource;

Statement (Resource, Resource, Resource, Resource) : Resource;

Kind (Statement\*) : Resource;

Class (Kind\*) : Resource;

Context (Class\*) : Resource;

Hierarchy: class (Object / Value) as superclass Context.

Object: class (extension);

Context: super class (intention);

(Kind, Statement, Role, Resource); Data (Resource Kind).

(Class, Kind, Statement, Role); Schema (Role Class)

(Context, Class, Kind, Statement); Interaction (Statement Context).

State Facet / Layer / Level / Augmentation / Model Resource Mappings.

Functional API: Message IO. Mappings.

State order (in context class hierarchies axes), comparison relations, iterations, flow, events, causal relations, units, enums, equivalence, etc.

Data order: Resource Kind hierarchies.

Schema order: Role Class hierarchies.

Interaction order: Statement Context hierarchies.

Meta Resource / Models / Messages: IDs / Encoding / Addressing formats. Ontology matching and Template / Augmentation / Transform enrichment (alignments), transforms (functors), materialization (model updates) via Mappings (events) and Meta Resource / Model Encoded Resource declarations (enrich / align, transform, updates algorithms: Encodings).

Meta Resources are used by a Model Meta Model for describing models. Some of them are:

URI

Resource

Context / Context

Subject / Occurrence

Predicate / Attribute

Object / Value

Statement

Model

Kind

ContextKind

SubjectKind

PredicateKind

ObjectKind

Message

Template

Augmentation

Transform

Class

Metaclass

Instance

(Augmentation, Template, Mapping, Transform); Meta Resources.

Model declared as Interaction Model Augmentation (matching Messages) in Interaction Model. Flows:

Message

Model (Functor)

Augmentation (Addressable Interaction)

Template (Message)

Mapping (Functor)

Transform (Message)

Model (Functor)

Message

Interaction Model: Model Events (Augmentation).

Augmentation: Event. Signature. Declarations / Occurrences. Domain, Input / Mapping, Transform / Range, Output.

Embeddings: Message Match Event Signature. Tempate matching / Transform rendering. Dataflow.

Embeddings: Meta Model Augmentation, Template, Mapping, Transform Meta Resources (input layer). Meta Model Source, Session, Interaction levels.

Model OntResource Augmented with Event Transform aggregates new Event Mapping.

**8: Resource APIs**

**8.1: Resource Monad**

Resource / Message Monad Events: Augmentations. Mapping: Endpoint. Events: Implement Message / Resource / URIs Protocols.

Monadic wrapper for which Augmentation (Functor Events) are declared into Interaction Model. Model(s) themselves are Augmrntation(s). Augmentation Statement Context Kind defines Event “signature”: Resource input / output Event domain / range. Output from an Event application (Transform) may feed back Model triggering further events (Dataflow). Augmentation Template, Mapping and Transform may behave as placeholder for Dataflow rendering of Meta Models.

Model Resources react to events according Message matching event “signature”.

**8.2: Functional APIs**

**8.3: Streams (Occurrences, etc.)**

**8.4: Transforms (Augmentation)**

**9: Messages**

Messages: Mappings. Meta Resources / Model Message based Model interactions (Subscriptions / Mappings).

Messages: Message semantics (Augmentation: Verbs, CRUD, Behavior) according Message structure / pattern (dialog / prompts).

Messages: Dataflow Template matches signatures (Session level, enrichs Message with Model / Dialog prompts / contents). Augmentation Functor applied over Message contents (Interaction level). Transform matching output signature emits (Session level, populated / prompts) output Message.

Messages: Dataflow. Subscriptions. Reactive Model. Dynamic subscriptions / bindings. Events publish / subscribe between Model Resource. Mappings.

Mappings: Declarative IO signatures: Context Kinds Templates / Transforms. Subscriptions / routes. Dataflow.

Messages: Mappings. Meta Resources / Model Message based Model interactions (Mappings : Subscriptions).

Messages: Message semantics (Augmentation: Verbs, CRUD, Behavior) according Message structure / pattern (dialog / prompts).

Messages: Dataflow Template matches signatures (Session level, enrichs Message with Model / Dialog prompts / content alignments). Augmentation Functor applied over Message contents (Interaction level). Transform matching output signature emits (Session level, populated / prompts) output Message.

Messages: Dataflow. Subscriptions. Reactive Model. Dynamic subscriptions / bindings. Events publish / subscribe between Model Resource. Mappings.

Augmentation Event input (Template) / output (Transform) declaration / instance (Mapping). Augmentation Mapping range declaration / result instance (Transform Message).

Augmentation: basic operation.

Resource Set Specification (Statement) matching Model which returns augmented Message response (Model I/O).

Augmentation declarative Model definitions.

Message Resolution Algorithm.

Protocol: Augmentation Message dialog I/O.

**9.1: Message Monad**

Models:

URI(s);

OntResource; Merged URI(s) wrapper.

Resource (OntResource CSPO / Contexts hierarchies Monad wrapper);

Message (Resource Monad wrapper); Request / Response Encoding.

Template / Transform (Message blueprints) domain / range : Message.

Augmentation : Functor.

Mappings: Declarative IO signatures: Context Kinds Templates / Transforms. Subscriptions / routes. Dataflow.

Encoding: Template Message augmentation (inputs).

Encoding: Declarative functors behavior encoding statements. Mappings (subscription / routes).

Encoding: Transform Message augmentation (outputs).

(Augmentation, Template, Mapping, Transform);

Dataflow: Order, Flows (Mappings, hierarchies).

Encode Model Events Interactions (Augmentation) inputs (Template) and outputs (Transform) wrapping corresponding Resource(s).

(Augmentation, Template, Mapping, Transform);

Model declared as Interaction Model Augmentation (matching Messages) in Interaction Model. Flows:

Message

Model (Functor)

Augmentation (Addressable Interaction)

Template (Message)

Mapping (Functor)

Transform (Message)

Model (Functor)

Message

Interaction Model: Model Events (Augmentation).

Augmentation: Event. Signature. Declarations / Occurrences. Domain, Input / Mapping, Transform / Range, Output.

Embeddings: Message Match Event Signature. Tempate matching / Transform rendering. Dataflow.

Embeddings: Meta Model Augmentation, Template, Mapping, Transform Meta Resources (input layer). Meta Model Source, Session, Interaction levels.

Model OntResource Augmented with Event Transform aggregates new Event Mapping.

Encoding: Template Message augmentation (inputs).

Encoding: Declarative functors behavior encoding statements. Mappings (subscription / routes).

Encoding: Transform Message augmentation (outputs).

(Augmentation, Template, Mapping, Transform);

Dataflow: Order, Flows (Mappings, hierarchies).

IDs: Addressing / Encoding. Semantic (signature, contents, context) resolvable / discoverable identifiers.

**9.2: Functional APIs**

**9.3: Streams (Signatures)**

**9.4: Transforms (Mapping)**

**9.5: Persistence**

Augmentations: Interaction Model Mappings execution / persistence / retrieval. Reactive model via representation of IDs: Mappings (signatures) dataflow inferred Augmentations.

Persistence: (activation / passivation): IDs / Meta Model / Facets from Interaction Model events (Messages) from Node IO. Interaction Model: Main Model(s) Message IO.

Messages: Saga Activation. Interaction Model (Meta Model). Aggregated (Interaction) Meta Model interactions (performed / inferred / possible) emitted as Model event Messages (Saga pattern). Mappings.

Saga Activation / Passivation populating Node local Quad store / persisting peers via DIDs ([ont.io](http://ont.io)) semantic (resolvable / discoverable) identifiers.

Messages: Saga Activation. Interaction Model (Meta Model). Aggregated (Interaction) Meta Model interactions (performed / inferred / possible) emitted as Model event Messages (Saga pattern). Mappings.

Messages: Saga Passivation. Model layers data routed by Mappings as event Message into (Interaction) Meta Model. Message inputs: Models. Mappings. Populate.

**10: IDs: Addressing / Encoding**

IDs:

URI(s);

OntResource; Merged URI(s) wrapper.

OntResource hierarchy: layers statement contexts. Facets DOM, Actor / Role.

Resource (OntResource Context Roles hierarchies Monad wrapper);

Statement : Resource quad, Resource.

Message (Resource Monad wrapper); Request / Response Encoding.

Template / Transform (Message blueprints) domain / range : Message.

IDs:

A: ID

B: Transform

C: Mapping

D: Template

E: Augmentation

F: Model

G: OntResource

H: CSPO Role

I: Statement

J: Kind

K: Class

L: Context

State Facet / Layer / Level / Augmentation / Model Resource Mappings.

Meta Resources layers contexts class hierarchy:

Context : Class : Kind : Statement : Role : Resource : Model : Augmentation : Template : Mapping : Transform : ID;

Meta Resources layers Reification: Context : Class : Kind : Statement : Role : Resource : Model : Augmentation : Template : Mapping : Transform as ID Context Statement (ID statements for each context layer). Model Levels (Facets / Levels). Aggregate reifications.

Meta Model:

A: (ID, ID, ID, ID);

B: (Transform, ID, ID, ID);

C: (Mapping, Transform, ID, ID);

D: (Template, Mapping, Transform, ID);

E: (Augmentation, Template, Mapping, Transform);

F: (Model, Augmentation, Template, Mapping);

G: (Resource, Model, Augmentation, Template);

H: (Role, Resource, Model, Augmentation);

I: (Statement, Role, Resource, Model);

J: (Kind, Statement, Role, Resource); Data (Resource Kind).

K: (Class, Kind, Statement, Role); Schema (Role Class)

L: (Context, Class, Kind, Statement); Interaction (Statement Context).

ID: (L (K (J (I (H (G (F (E (D (C (B (A, Nil))))))))))));

Cons lists. Binary Trees. Huffman / Prefix codes. RDF List serialization. Meta Resources / Models declarative statements Encoding, Addressing.

Kinds, Signatures. Contents. Contextual metadata.Lattices. Roles.Sets (bitstring cuads). Definitions (elements). Operations. Rules. Categories. Groups.

ID: Augmentation occurrences metadata in Statement contexts;

Statement: (ID (ID (ID (ID, Nil))));

(C (S (P (O, Nil))));

IDs, Meta Model, Interaction Model (Level), Session (Level), Backend (Level), Facets features:

Augmentations: Interaction Model Mappings execution / persistence / retrieval. Reactive model via representation of IDs: Mappings (signatures) dataflow inferred Augmentations.

Persistence: (activation / passivation): IDs / Meta Model / Facets from Interaction Model events (Messages) from Node IO. Interaction Model: Main Model(s) Message IO.

Order / comparisons: tree representation ordered by Context Role class hierarchy, instances hierarchies and aggregation hierarchies. Resources order (IDs). Statements order (Statement IDs). Comparison criteria (choose relevant IDs). ToDo.

Semantic resolution: Query Resource(s) satisfying “criteria” (i.e.: Object(s) for predicate) IDs by IDs resolution pattern:

Query Resources by role in context.

Query Resources by attributes / values.

Query Resources by identity / type.

Message (Resource Monad wrapper) : Statement; Request / Response Encoding.

Template / Transform (Message blueprints) domain / range : Message.

Augmentation : Functor.

Mappings: Immutable Resources Message based Augmentation bindings. Dataflow subscription routes: Signatures / CKs (Augmentation(s) functional streams).

Subject Kind: Subjects stream. Object Kind: Objects stream.

Encoding: Template Message augmentation (inputs).

Encoding: Declarative functors (Augmentation) behavior encoded in statements. Mappings (subscription / routes).

Encoding: Transform Message augmentation (outputs).

Interaction Model (Interaction Level):

(Augmentation, Template, Mapping, Transform);

Dataflow: Order, Flows (Mappings, hierarchies).

Interaction Model: aggregated Meta Model interactions (performed / inferred / possible) declared Models events (saga pattern).

Messages: Saga Activation. Interaction Model (Meta Model). Aggregated (Interaction) Meta Model interactions (performed / inferred / possible) emitted as Model event Messages (Saga pattern). Mappings.

Message - Model - Template (data : Resource) - Augmentation (functor) - Transform (interaction : Resourcr) - Model - Message.

Encoding, IDs: magic numbers (MIME types : Context Kinds), metaclass, class, instance, context, CSPO, etc. relations "contextual slots" for IDs. Resource resolution, Operation (primes, encoded lattice, slots context relations) factors in Meta Model relations. Encode order, hierarchies, temporal, causal (reified), containment, etc. relations into IDs encoding. Ontology matching: encoded IDs roles in context aggregation / learning.

Explain URI, Resource, Layers, Model, Kinds, etc. APIs. Meta Resources. Meta Model. Hierarchies. Order. Iteration. Flows.

Model, URIs, Resource, Contexts Functional APIs. Meta Model / Resources encoding. Mappings.

IDs: Addressing / Encoding. Semantic (signature, contents, context) resolvable / discoverable identifiers.

Ontology Matching:

Ontology Matching: IDs, Addressing, Encoding. Functional, Semiotic, Dimensional (Facets). Layers. Levels. Meta Resource / Model. Sets. Value as occurrence of attribute. metaclass / class / instance IDs.

Ontology Matching: Encode: order, iteration, flows, units, relations, events, enums, etc.

Messages CRUD / Invocation semantics. Dialog. Prompts.

Encoding: Cons lists. Trees. Huffman / Prefix codes. RDF List serialization. Meta Resources / Models declarative Encoding, Addressing, Mappings, Transforms (Immutable sequences, dataflow Mapping: Template / Augmentation / Transform functional streams).

Kinds, Signatures. Contents. Contextual metadata. Sets (bitstring cuads). Lattices.

(C (S (P (O, Nil))));

(C2 (C (S (P, Nil)));

Models:

URI(s);

OntResource; Merged URI(s) wrapper.

Resource (OntResource CSPO / Contexts hierarchies Monad wrapper);

Message (Resource Monad wrapper); Request / Response Encoding.

Template / Transform (Message blueprints) domain / range : Message.

Augmentation : Functor.

Mappings: Declarative IO signatures: Context Kinds Templates / Transforms. Subscriptions / routes. Dataflow.

Encoding: Template Message augmentation (inputs).

Encoding: Declarative functors behavior encoding statements. Mappings (subscription / routes).

Encoding: Transform Message augmentation (outputs).

(Augmentation, Template, Mapping, Transform);

Dataflow: Order, Flows (Mappings, hierarchies).

Message (Resource Monad wrapper) : Statement; Request / Response Encoding.

Template / Transform (Message blueprints) domain / range : Message.

Augmentation : Functor.

Mappings: Immutable Resources Message based Augmentation bindings. Dataflow subscription routes: Signatures / CKs (Augmentation(s) functional streams).

Subject Kind: Subjects stream. Object Kind: Objects stream.

Encoding: Template Message augmentation (inputs).

Encoding: Declarative functors (Augmentation) behavior encoded in statements. Mappings (subscription / routes).

Encoding: Transform Message augmentation (outputs).

Interaction Model (Interaction Level):

(Augmentation, Template, Mapping, Transform);

Dataflow: Order, Flows (Mappings, hierarchies).

IDs:

URI(s);

OntResource; Merged URI(s) wrapper.

OntResource hierarchy: layers statement contexts. Facets DOM, Actor / Role.

Resource (OntResource Context Roles hierarchies Monad wrapper);

Statement : Resource quad, Resource.

Message (Resource Monad wrapper); Request / Response Encoding.

Template / Transform (Message blueprints) domain / range : Message.

IDs:

A: OntResource.

B: CSPO Role.

C: Statement : OntResource Occurrence.

D: Kind CSPO Instances.

E: Class : Kind CSPO Classes.

F: ContextStatement : Context Role.

Meta Model:

A: (Resource, ?, ?, ?);

B: (Role, Resource, ?, ?);

C: (Statement, Role, Resource, ?);

D: (Kind, Statement, Role, Resource); Data (Resource Kind).

E: (Class, Kind, Statement, Role); Schema (Role Class)

F:.(ContextStatement, Class, Kind, Statement); Interaction (Statement ContextStatement).

ID: (F (E (D (C (B (A, Nil))))));

Cons lists. Binary Trees. Huffman / Prefix codes. RDF List serialization. Meta Resources / Models declarative statements Encoding, Addressing.

Kinds, Signatures. Contents. Contextual metadata.Lattices. Roles.Sets (bitstring cuads). Definitions (elements). Operations. Rules. Categories. Groups.

Statement: (ID (ID (ID (ID, Nil))));

(C (S (P (O, Nil))));

Order / comparisons: tree representation ordered by Context Role class hierarchy, instances hierarchies and aggregation hierarchies. Resources order (IDs). Statements order (Statement IDs). Comparison criteria (choose relevant IDs). ToDo.

Semantic resolution: Query Resource(s) satisfying “criteria” (i.e.: Object(s) for predicate) IDs by IDs resolution pattern:

Query Resources by role in context.

Query Resources by attributes / values.

Query Resources by identity / type.

Encoding: Resource ID. Encoded Resource contents (signature / occurrence). Augmentation: Resource set (Message) resolution from context over Template / Resource(s).

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).

Encode common upper Semiotic / Dimensional Model: Reference Model.

Encode Kind / Context hietarchies.

Encode Augmentation(s) as Resource descriptions.

Encode Model(s) as Respurce set. Meta Resources, layers Contexts, Kinds (reified).

Encode Graph Execution Semantics. Dataflow: Context Kind signatures. Iteration, conditional jumps.

Events / Messaging.

URIs, metaclass, class, instance, context, occurrence IDs. Formulae.

Resources wraps URIs streams sources / sinks activated by ontology matching alignment. Aggregates same entity different URIs, representations in contexts.

Context Kind / Signature: Predicate Kind from Subject / Object Kind.

Object occurrence of Predicate.

Encode behavior: iteration / jumps. Order statements (URIs APIs).

Meta Resource(s): Resources / Messages reifying "patterns" on inputs (URI, Resource, Statement, Kind(s), Context, Occurrence, Attribute, Value, Layer Context classes, etc.). Declarative statement for Augmentation shapes applyied to input contexts.

Sets. Quads.

Metaclass / Class / Instance.

Class / Instance ID pairs:

Subject / Context / Role : Attribute, Value. Metamodel. Encoding: each type as each (pair) kind. Pairs.

Semiotic encoding:

(Context, Sign, Concept, Object);

Value as Occurrence of Attribute in Attribute Occurrence Context. Meta Resource context roles).

Augmentation. Transform. Backend. DIDs: events sourcing (decentralized persistence). Encoding: avoid / resolve duplicate transactions.

Encoding: Resource ID. Encoded Resource contents (signature / occurrence). Augmentation: Resource set (Message) resolution from context over Template / Resource(s).

Augmentation: Message signature matches Template signature (across types hierarchies): Transform results Resource(s) for Augmentation predicates / mappings. Mappings: Meta Resources, Patterns,  Augmentations (in contexts), common hierarchy super Resource. Variables, expressions.

Message - Model - Template (data) - Augmentation (functor) - Transform (interaction) - Model - Message.

Encoding, IDs: magic numbers (MIME types : Context Kinds), metaclass, class, instance, context, CSPO, etc. relations "contextual slots" for IDs. Resource resolution, Operation (primes, encoded lattice, slots context relations) factors in Meta Model relations. Encode order, hierarchies, temporal, causal (reified), containment, etc. relations into IDs encoding. Ontology matching: encoded IDs roles in context aggregation / learning.

Explain URI, Resource, Layers, Model, Kinds, etc. APIs. Meta Resources. Meta Model. Hierarchies. Order. Iteration. Flows.

Model, URIs, Resource, Contexts Functional APIs. Meta Model / Resources encoding. Mappings.

IDs: Addressing / Encoding. Semantic (signature, contents, context) resolvable / discoverable identifiers.

Ontology Matching:

Ontology Matching: IDs, Addressing, Encoding. Functional, Semiotic, Dimensional (Facets). Layers. Levels. Meta Resource / Model. Sets. Value as occurrence of attribute. metaclass / class / instance IDs.

Ontology Matching: Encode: order, iteration, flows, units, relations, events, enums, etc.

Messages CRUD / Invocation semantics. Dialog. Prompts.

Encoding: Cons lists. Trees. Huffman / Prefix codes. RDF List serialization. Meta Resources / Models declarative Encoding, Addressing, Mappings, Transforms (Immutable sequences, dataflow Mapping: Template / Augmentation / Transform functional streams).

Kinds, Signatures. Contents. Contextual metadata. Sets (bitstring cuads). Lattices.

(C (S (P (O, Nil))));

(C2 (C (S (P, Nil)));

Models:

URI(s);

OntResource; Merged URI(s) wrapper.

Resource (OntResource CSPO / Contexts hierarchies Monad wrapper);

Message (Resource Monad wrapper); Request / Response Encoding.

Template / Transform (Message blueprints) domain / range : Message.

Augmentation : Functor.

Mappings: Declarative IO signatures: Context Kinds Templates / Transforms. Subscriptions / routes. Dataflow.

Encoding: Template Message augmentation (inputs).

Encoding: Declarative functors behavior encoding statements. Mappings (subscription / routes).

Encoding: Transform Message augmentation (outputs).

(Augmentation, Template, Mapping, Transform);

Dataflow: Order, Flows (Mappings, hierarchies).

**11: Dataflow**

Messages: Dataflow Template matches signatures (Session level, enrichs Message with Model / Dialog prompts / content alignments). Augmentation Functor applied over Message contents (Interaction level). Transform matching output signature emits (Session level, populated / prompts) output Message.

Messages: Dataflow. Subscriptions. Reactive Model. Dynamic subscriptions / bindings. Events publish / subscribe between Model Resource. Mappings.

Interaction Model declares Events (Augmentations) which have a functional Mapping between its domain (Template) and range (Transform). An Augmentation Context Kind correspond to this Mapping “signature”. Dataflow binds input Message(s) to domain Template by pattern matching and resolving any input Message references (Addressing).

Outputs are resolved by pattern matching with Transform, Message and existing Model data. Augmentations may play the role of “placeholder” Resource(s) which are bound to context aware Augmentations thus rendering Transforms into Model entities (including Mapping Augmentations themselves).

Model declared as Interaction Model Augmentation (matching Mappings) in Interaction Model. Flows:

Message

Model (Functor)

Augmentation (Addressable Interaction)

Template (Message)

Mapping (Functor)

Transform (Message)

Model (Functor)

Message

Interaction Model: Model Events (Augmentation).

Augmentation: Event. Signature. Declarations / Occurrences. Domain, Input / Mapping, Transform / Range, Output.

Embeddings: Message Match Event Signature. Tempate matching / Transform rendering. Dataflow.

Embeddings: Meta Model Augmentation, Template, Mapping, Transform Meta Resources (input layer). Meta Model Source, Session, Interaction levels.

Model OntResource Augmented with Event Transform aggregates new Event Mapping.

Explain Context layers Aggregation Augmentation. Example: Role(s) for each CSPO. Entity in Statements. Meta Model. Meta Resources.

Explain Context layers Alignment Augmentation. Meta Model. Meta Resources.

Explain Context layers Activation Augmentation. Meta Model. Meta Resources.

Explain Augmentation. Context : Functor<Template, Transform>; Template, Transform : Context Kind (Levels: Data, Session, Interaction / Functor instance / execution contexts?).

Functors: Meta Model declarations / Context classes / instance declarative implementations. Aggregation type: invocation over each CSPO / Context roles.

Functor applied to context: Aggregation.

Functor applied to subject: Alignment.

Functor applied to predicate: Activation.

Functor applied to object: members traversal.

Message. For each layer perform each Functor: (Object : aggreg, Kind : activ, Attr : align, Obj : onto).

Augmentation:

Functors: Augmentation declaration: Meta Model definitions (Context class / instances). Message: dataflow matches Template signatures: interactions. Apply Augmentation Functors over Message contents (interactions enrich Message with Models contents: ontology matching / Levels / Facets). Materialize / emit dialog / prompts Message (enrich Message from Models / reactive IO events).

Augmentation: Context / Functors. Message Resource(s) / Meta Resource(s) (nested / wrapped) elements determines flow Template Transform results / behaviors (CRUD, Functor invocations). Message IO performs Augmentations. Ontology levels resolution (Templates / Transforms / Augmentatiom levels: matching patterns / dialog prompts in Ontology levels).

Interactions declarations: signature definitions (Template / Transform contexts). Interaction instances: addressable exchanges (Augmentations, Message, Model context / Mapping bindings / matchings / performances). Meta Model / Levels event driven Model Augmentation.

Interactions declarations: signature definitions (Template / Transform contexts). Interaction instances: Exchanges (Augmentations, Message, Model context / Mapping bindings / matchings / performance). Contexts / Exchanges: Meta Model / Levels event driven source Augmentation events declarations (populating Facets / Layers / Levels).

Models:

URI(s);

OntResource; Merged URI(s) wrapper.

Resource (OntResource CSPO / Contexts hierarchies Monad wrapper);

Message (Resource Monad wrapper); Request / Response Encoding.

Template / Transform (Message blueprints) domain / range : Message.

Augmentation : Functor.

Mappings: Declarative IO signatures: Context Kinds Templates / Transforms. Subscriptions / routes. Dataflow.

Encoding: Template Message augmentation (inputs).

Encoding: Declarative functors behavior encoding statements. Mappings (subscription / routes).

Encoding: Transform Message augmentation (outputs).

(Augmentation, Template, Mapping, Transform);

Dataflow: Order, Flows (Mappings, hierarchies).

**11.1: Dataflow (Streams)**

Signatures: Routes (bindings).

Augmentations: Transforms.

Message

Model

Resource

Model

Message

**11.2: Model Interaction Layer matches Messages**

**11.3: Interaction Layer (Augmentation Dataflow)**

**11.4: Meta Resources (Template, Transform) / Transform dataflow embeddings**

**11.5: Signatures. Addressing / Discovery. Bindings.**

**12: Augmentation (Transforms)**

Augmentations: matching Events Functors aggregate / align / activate (classify) sources of ontology matched data / schema / behavior enabling semantic layers interoperation.

Interaction Model Event. Matches Message signature (domain Template / range Transform) performing Mapping. Dataflow: Transform output matches another Event signature. Embedding: OntResource augmented with new referenced aligned / matched Model entity.

Augmentations defined as declarative Mappings in Interaction Model encoding Context (layer) inputs matching signatures and augments current / previous layer emmiting mapping transforms.

Functors: Meta Model declarations / Context classes / instance declarative implementations. Aggregation type: invocation over each CSPO / Context roles.

Functor applied to context: Aggregation.

Functor applied to subject: Alignment.

Functor applied to predicate: Activation.

Functor applied to object: members traversal.

Reactive Context Kind (matching signatures) dataflow.

Message - Model - Template (context) - Augmentation (interaction) - Transform (data) - Model - Message.

Implementation API: Node / Container. Services (URIs Context Kind signatures resolution).

Core Services: Activation Augmentation (Naming).

Core Services: Alignment Augmentation (Index).

Core Services: Aggregation Augmentation (Registry).

Core Services: RDF / OWL Backend (endpoint, reasoning, persistence).

Core Services: DIDs Persistence (sync Node state: events sourcing).

Core Services: Protocol (I/O). Node, Session, Intetaction levels. Base Connector Augmentation API. Event driven URIs dialog / prompts protocol adapters.

Explain Context layers Aggregation Augmentation. Example: Role(s) for each CSPO. Entity in Statements. Meta Model. Meta Resources.

Explain Context layers Alignment Augmentation. Meta Model. Meta Resources.

Explain Context layers Activation Augmentation. Meta Model. Meta Resources.

Explain Augmentation. Context : Functor<Template, Transform>; Template, Transform : Context Kind (Levels: Data, Session, Interaction / Functor instance / execution contexts?).

Functors: Meta Model declarations / Context classes / instance declarative implementations. Aggregation type: invocation over each CSPO / Context roles.

Functor applied to context: Aggregation.

Functor applied to subject: Alignment.

Functor applied to predicate: Activation.

Functor applied to object: members traversal.

Message. For each layer perform each Functor: (Object : aggreg, Kind : activ, Attr : align, Obj : onto).

Augmentation:

Functors: Augmentation declaration: Meta Model definitions (Context class / instances). Message: dataflow matches Template signatures: interactions. Apply Augmentation Functors over Message contents (interactions enrich Message with Models contents: ontology matching / Levels / Facets). Materialize / emit dialog / prompts Message (enrich Message from Models / reactive IO events).

Augmentation: Context / Functors. Message Resource(s) / Meta Resource(s) (nested / wrapped) elements determines flow Template Transform results / behaviors (CRUD, Functor invocations). Message IO performs Augmentations. Ontology levels resolution (Templates / Transforms / Augmentatiom levels: matching patterns / dialog prompts in Ontology levels).

Augmentation:

Functors: Meta Model declarations / Context classes / instance declarative implementations. Aggregation type: invocation over each CSPO / Context roles.

Functor applied to context: Aggregation.

Functor applied to subject: Alignment.

Functor applied to predicate: Activation.

Functor applied to object: members traversal.

Reactive Context Kind (matching signatures) dataflow.

Message - Model - Template (context) - Augmentation (interaction) - Transform (data) - Model - Message.

Implementation API: Node / Container. Services (URIs Context Kind signatures resolution).

Core Services: Activation Augmentation (Naming).

Core Services: Alignment Augmentation (Index).

Core Services: Aggregation Augmentation (Registry).

Core Services: RDF / OWL Backend (endpoint, reasoning, persistence).

Core Services: DIDs Persistence (sync Node state: events sourcing).

Core Services: Protocol (I/O). Node, Session, Intetaction levels. Base Connector Augmentation API. Event driven URIs dialog / prompts protocol adapters.

Explain Context layers Aggregation Augmentation. Example: Role(s) for each CSPO. Entity in Statements. Meta Model. Meta Resources.

Explain Context layers Alignment Augmentation. Meta Model. Meta Resources.

Explain Context layers Activation Augmentation. Meta Model. Meta Resources.

Explain Augmentation. Context : Functor<Template, Transform>; Template, Transform : Context Kind (Levels: Data, Session, Interaction / Functor instance / execution contexts?).

Functors: Meta Model declarations / Context classes / instance declarative implementations. Aggregation type: invocation over each CSPO / Context roles.

Functor applied to context: Aggregation.

Functor applied to subject: Alignment.

Functor applied to predicate: Activation.

Functor applied to object: members traversal.

Message. For each layer perform each Functor: (Object : aggreg, Kind : activ, Attr : align, Obj : onto).

Augmentation:

Functors: Augmentation declaration: Meta Model definitions (Context class / instances). Message: dataflow matches Template signatures: interactions. Apply Augmentation Functors over Message contents (interactions enrich Message with Models contents: ontology matching / Levels / Facets). Materialize / emit dialog / prompts Message (enrich Message from Models / reactive IO events).

Augmentation: Context / Functors. Message Resource(s) / Meta Resource(s) (nested / wrapped) elements determines flow Template Transform results / behaviors (CRUD, Functor invocations). Message IO performs Augmentations. Ontology levels resolution (Templates / Transforms / Augmentatiom levels: matching patterns / dialog prompts in Ontology levels).

**12.1: Augmentation Functor(Message, Message) : Events**

Augmentations: matching Events Functors aggregate / align / activate (classify) sources of ontology matched data / schema / behavior enabling semantic layers interoperation.

Aggregation: Infer input data streams data, schema, behavior class / instance context layers.

Alignment: Infer layer missing / deducible attributes and values.

Activation: Infer layer CSPO Kind / Roles. Basic type system.

Aggregation (Augmentation): Apply each Context (layer) Functor on inputs (from input layer) and emits Transform, matching corresponding (next) layer. Next layer Context and SPO according functional mapping declared by Meta Resource types on augmented layer.

Alignment (Augmentation): ToDo.

Activation (Augmentation): ToDo.

**12.2: Augmentation Functional APIs**

**12.3: Streams (Dataflow Model Events declaration)**

**12.4: Meta Resources: Interaction IO (Messages Meta Resources bindings / embeddings / prompts / mappings)**

**13: Ontology Matching**

Ontology matching. Dataflow: sort statements. Units. Equivalences. Distances / events (order). Services (Augmentation / Context Functors Meta Model mappings / transforms).

Explain ontology matching: data, schema, behavior alignments. Layers. Levels. Facets. Meta Resources / Model. IDs, Encoding / Addressing.

Encoding. Functional, Semiotic, Dimensional (Facets). Layers. Levels. Meta Resource / Model. Sets. Value as occurrence of attribute. metaclass / class / instance IDs.

Functionsl / Semiotic / Dimensional layers / levels examples / alignments.

Model, URIs, Resource, Contexts Functional APIs. Meta Model / Resources encoding. Mappings.

IDs: Addressing / Encoding. Semantic (signature, contents, context) resolvable / discoverable identifiers.

Ontology Matching:

Ontology Matching: IDs, Addressing, Encoding. Functional, Semiotic, Dimensional (Facets). Layers. Levels. Meta Resource / Model. Sets. Value as occurrence of attribute. metaclass / class / instance IDs.

Ontology Matching: Encode: order, iteration, flows, units, relations, events, enums, etc.

Ontology matching (Data, Schema, Behavior alignments):

Data alignment:

Determine if two instances (example: records) of two different backends or services refer to the same entity (Customers : John D. / Employees : John Doe).

Schema alignment:

Determine, for example, meaning and equivalences between diverse (aggregated / composite) schemas (equivalent classes / tables, equivalent attributes / columns, equivalent roles / relations).

Behavior alignment:

Determine meaning and equivalences between (aggregated / composite) behavior contexts and behavior contexts invocations / interactions (Appointment / Interview, anAppointment / anInterview. Behavior flows aggregated from backends / services learning).

Ontology Matching: IDs, Addressing, Encoding. Functional, Semiotic, Dimensional (Facets). Layers. Levels. Meta Resource / Model. Sets. Value as occurrence of attribute. metaclass / class / instance IDs.

Ontology Matching: Encode: order, iteration, flows, units, relations, events, enums, etc.

Semiotic / Dimensional alignment. TBD.

Ontology Matching. Semiotic. Dimensional. Sets. Functional Reference Model.

Semiotic / Dimensional alignment / aggregation layers (lower resource alignment layers):

(Context, Sign, Concept, Object);

(Value, Distance, Prev, Next : in Units); (Measure, Value...) (Unit, Measure, Value,...); (Resource, Unit, Measure, Value); Marriage example.

Messaging metamodel:

(Message, Resource, LHS, RHS);
(Interaction, Message, Resource, LHS);
(Role, Interaction, Message, Resource);
(Context, Role, Interaction, Message);
(Dataflow, Context, Role, Interaction);

Meta Model (Meta Resources)

Semiotic / Dimensional (encode matching Resources). Common upper ontology matching layers. Models:

Source Model. Data.

Grammar Model. Schema.

Interaction Model: Behavior?

Ontology matching (table, pk, col, val example). Helper upper models for models linking / alignment.

Meta Resource(s): URI, Resource, Statement, Model, CSPO, Layer, Context, Occurrence, Attribute, Value, Kind, etc.

Semiotic encoding:

(Context, Sign, Concept, Object);

Object as Sign: Concept: Attribute. Other mappings (roles).

Semiotic / Dimensional Alignment, Aggregation (known mappings) : Class / ID Ontology Matching. Contextual IDs (infer occurrence contexts). Inference ID lookup of ID for desired satisfaction of given transforms / roles / operations.

Ontology matching. Dataflow: sort statements. Units. Equivalences. Distances / events (order). Services (Augmentation / Context Functors Meta Model mappings / transforms).

Explain ontology matching: data, schema, behavior alignments. Layers. Levels. Facets. Meta Resources / Model. IDs, Encoding / Addressing.

Encoding. Functional, Semiotic, Dimensional (Facets). Layers. Levels. Meta Resource / Model. Sets. Value as occurrence of attribute. metaclass / class / instance IDs.

Functionsl / Semiotic / Dimensional layers / levels examples / alignments.

Ontology matching (Data, Schema, Behavior alignments):

Data alignment:

Determine if two instances (example: records) of two different backends or services refer to the same entity (Customers : John D. / Employees : John Doe).

Schema alignment:

Determine, for example, meaning and equivalences between diverse (aggregated / composite) schemas (equivalent classes / tables, equivalent attributes / columns, equivalent roles / relations).

Behavior alignment:

Determine meaning and equivalences between (aggregated / composite) behavior contexts and behavior contexts invocations / interactions (Appointment / Interview, anAppointment / anInterview. Behavior flows aggregated from backends / services learning).

Model, URIs, Resource, Contexts Functional APIs. Meta Model / Resources encoding. Mappings.

IDs: Addressing / Encoding. Semantic (signature, contents, context) resolvable / discoverable identifiers.

Ontology Matching:

Ontology Matching: IDs, Addressing, Encoding. Functional, Semiotic, Dimensional (Facets). Layers. Levels. Meta Resource / Model. Sets. Value as occurrence of attribute. metaclass / class / instance IDs.

Ontology Matching: Encode: order, iteration, flows, units, relations, events, enums, etc.

**14: Deployment**

**14.1: Bus**

**14.2: Service**

**14.3: Connector**

**15: Services**

Services. Connectors. URIs APIs. Endpoints (Events Mapping) messaging interface.

**15.1: Registry**

**15.2: Index**

**15.3: Naming**

**16: Default Augmentations**

**16.1: Aggregation**

**16.2: Alignment**

**16.3: Activation**

**17: Implementation**

Persistence

Messaging

Bus

Service

Connector

**18: Client Platforms**

Connector Activation.

**19. To Do**

Augmentation: Aggregation, Alignment, Activation.

Meta Model. Facets. Encoding. Augmentation: Functional, Semiotic, Dimensional inference / learning.

Ontology Matching: cross Facets Augmentation over Reference / Meta Model.

Model Facets: Predictions. Meta Model Facets possible Augmentation(s).

Map Reduce: Augmentation. Mapping (map) Transform. Streams (routes bindings). Template key, Transform value.

Encoding: Occurence(s) Context awareness. Kinds / Roles. Meta Model / Augmentation / IDs.

Model APIs. Monadic / Functional:

Dataflow: Mappings. Augmentations. Streams: Monadic chaining from chained sources / events. Signatures (Context Kind) domain / range route bindings.

Monad: Quad. Reified CSPO Statement: (Context, Occurrence, Attribute, Value); Parent Augmentation Monad.

Layers (Aggregation fmap):

(Augmentation, Model) : Resource.

(Model, Resource) : Role.

(Resource, Role) : Statement.

(Role, Statement) : Kind.

(Statement, Kind) : Class.

(Kind, Class) : Flow.

(Class, Flow) : Behavior.

Augmentation (Alignments fmap):

(Model, Augmentation) : Template.

(Augmentation, Template) : Mapping.

(Template, Mapping) : Transform.

(Resource, Model) : Augmentation.

(Role, Resource) : Model.

(Statement, Role) : Resource.

(Kind, Statement) : Role.

(Class, Kind) : Statement.

(Context, Class) : Kind.

Types: Roles (CSPO) Kinds (Activation fmap):

(Statement, Role) : Kind. TBD.

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.

Deployment:

Jersey / CDI. NodeJS. Others.

Protocol (Events / Encoding : Augmentation / Model). APIs / Services interface. Node: Signature routes bindings. Bus. Message I/O: Model objects parsing / rendering (Augmentation / Backend).

Models. Functional object models APIs. Messages (Augmentations, Models. Dialog: Protocol).

APIs: Protocol based declarative Model interaction interfaces. Endpoints: CRUD, ETL, Connectors, Clients, OGM / DOM. Domains activation modelling.

Services: Protocol based distributed declarative interaction interfaces. Protocol integration layer.

Naming: Resolve distributed names / matching identifiers contexts. Aggregation.

Index: Search distributed identifiers by attributes. Augmentation. Alignment.

Registry: Bound identifier context attributes / lookup identifiers roles. Activation.

Backend: Model Indices (Context layers, Statement Roles, IDs): combinations / permutations / rotations. Filter (Map Reduce queries: match patterns). Augmentations. Quads: rotations, index layers occurrences. Encoding IDs: tuple(s), permutations. Permutations / rotations: prefix permutations of indexed tuple.

Backend: Dataflow index. Index Augmentation(s) / Mapping domain / range signatures patterns ordered bindings (streams / routes). Mapping Rules: Grammar, Combination transforms. Order. Repetition.

Backend Quads (layers): Augmentation (Context) input given to Template (Subject) : Mappings.

Backend Quads (layers): Model (Context) input given to Augmentation (Subject) : Templates.

Backend Quads (layers): Resource (Context) input given to Model (Subject) : Augmentations (Resource is result of Augmentation instance in Model).