**Contents / Features (Mision / Vision). Distributed consistent Knowledge Applications.**

**RDF / OWL, Graphs, Triples, Quads introduction.**

TBD.

**Model: Object Graph Representation as RDF Quads.**

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 types and values for the aggregated (S) Object of Class (C).

Contexts. Occurrences, Attributes, Values:

Subject in Statement has Predicate and Object Attribute / Value.

Predicate in Statement has Subject and Object Attribute / Value.

Object in Statement has Subject and Predicate Attribute / Value.

Value as Occurrence of Attribute in Attribute Occurrence Context.

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

Instance, occurrence, class, metaclass.

**URIs, Resource, Statement, Layer, Kind APIs.**

Augmentation: basic operation.

RDF Backend. URIs Services.

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.

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.

Context Kind Signatures.

Datasources / Backends / Services.

Ontology matching (Backend / Interaction Model).

Model Meta Resource: Model components reified Resource types / instances (URIs, Resource, Statement, Context : Layer, Kind, etc.). Augmentation templates "placeholders".

Kinds:

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.

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).

**Functional Implementation: URI / Resource APIs.**

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).

**Model Layers:**

Augmentation: basic operation.

Layered data, schema, behavior class / instance quads hierarchy. Model layers: URI quads:

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);

Graph Execution Semantics.

Ontology Matching. Upper ontologies. Primitives.

**Model I/O Dataflow:**

Augmentation: basic operation.

Layers (declaratively stated in Interaction Model):

Data input statements (Message).

Aggregate layers.

Align attributes.

Activate Kind.

Model: Reactive entity applying Message Augmentation resolving Resource Set Specification Message from inputs. Data Message (URIs layer), dataflow Message (Model / dialog).

Message Resolution Algorithm.

Data instance inputs (URIs events).

Model Message Augmentation resolution.

Interaction Model events / distributed / inference sourcing. Augmentations / CRUD: Interaction Model DIDs. URIs quad store / backend.

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

**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.

**Interaction Model:**

Augmentation: basic operation.

Source (upper) Model. Models hierarchies aligned with Interaction Model.

Interaction Model provides event sourcing, distributed inference / synchronization (distributed consolidation and alignments).

Interaction Model I/O : Message (from URIs or events) perform and materialize applying Augmentation from Interaction Model population.

Message declaratively states Model Specification through Message Augmentations.

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.

Augmentations:

Data (Message, Aggregation);

Schema (Alignment, Activation);

Behavior (Transform, Specification);

Interaction Model Specification (Metacircular interpreter: encodes Model(s), including itself): Interaction Model reifies / declaratively renders Source, Metagraph, Dimensional, Grammar Models via Augmentation Specification Message(s) from which it is populated and to which Augmentation (input Message) is performed, populating corresponding Model Resource(s).

Functional (monadic) Message Resolution Algorithm. Encoding.

**Augmentation:**

Augmentation: basic operation.

Augmentation: Basic Model I/O operation. Apply Model / Service (layers dataflow) to input Message quads. Layer. Dialog.

Messages Resource Set Specifications for CRUD, Aggregation, Alignment, Activation over Model. (Interaction Model Specification) stated on Interaction Model or from Protocol Message.

Model I/O: Augmentation Message application over Model from backend (URIs) Message or from Model I/O (layers) Message. Returns Resource Set populated / materialized Message.

Model I/O: layers application. Output model layers classes (layer Context) as stated in Interaction Model for input Message.

Model I/O: application of layer context class, state context, occurrence, attribute, etc. placeholders (value of placeholer in inputs) via reified statement roles in CSPO of layer statement specification (output).

Augmentation state Occurrence aggregation of Attribute / Values (i.e.: Statement / Roles), CSPO rendering / translation to output Message and transforms as specified in Intetaction Model.

Augmentation: each Augmentation populates corresponding Models performing CRUD, aggregation, inference and classification augmentations from Interaction Model Specification.

Layers. Augmentation: new IDs / ID Contexts. Naming.

**CRUD (I/O Message) Augmentation:**

Augmentation: CRUD (I/O Message).

Specification Model: Source.

Augmented Models (materialize, aggregate, align, activate).

**Aggregation Augmentation:**

Augmentation: Context Aggregation. Specification Model: Metagraph. Classification (aggregate quads contexts context / roles / class / identity).

**Alignment Augmentation:**

Augmentation: Data Alignment. Specification Model: Dimensional. Clustering (inference of links / attributes).

**Activation Augmentation:**

Augmentation: Interaction Activation. Specification Model: Grammar. Regression (classify roles in contexts: Kind).

**Models:**

Models hierarchies aligned with Interaction Model. Source, Metagraph, Dimensional, Grammar.

**Interaction Model Specification.**

Message / Aggregation (data)

Alignment / Activation (schema)

Transform / Specification (behavior).

Align to: URIs, Resource, Statement, Kind, Context Kind, Context, Occurrence, Attribute, Value.

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

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);

Statement Aggregation: Statement instance Context for each distinct CSPO URI on inputs aggregates same URI Occurrence as Subject with corresponding Attribute (output Predicate) / Value (output Object). According CSPO input as Occurrence, corresponding Attributes / Values are chosen.

Resource : Functional URI wrapper.

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

(Statement, Occurrence, Attribute, Value);

Aggregation layer: for each previous layer Message, layers: (Aggregation Instance, previous Message Context as Subject, previous Message S/P as Attribute / Value). Previous layer: Aggregation until end of source Messages layers (6 Aggregation statements consuming previous CSPOs. Renders to Aggregation instance contexts of Aggregation class).

Alignment layer: Context / Occurrence / Attribute / Value. Renders augmented Attribute / Value Context / Occurrence.

Activation layer: for each layer Message, Activation (Kind instances) are for each Activation class taking one of Message CSPO as Kind Subject and their corresponding CSPOs as Attribute / Value. Kind classes for each Aggregation layer. Context Kind: composite Subject / Predicate Kinds as Attribute / Value.

Layers dataflow: hierarchical Message inputs / outputs.

**Source Model Specification.**

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);

**Metagraph Model Specification.**

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);

**Dimensional Model Specification.**

(Value, Previous, Distance, Next);  
(Measure, Value, Previous, Distance);  
(Unit, Measure, Value, Previous);  
(Dimension, Unit, Measure, Value);  
(Concept, Dimension, Unit, Measure);  
(Resource, Concept, Dimension, Unit);  
(Statement, Resource, Concept, Dimension);

Example:

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

**Grammar Model Specification.**

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);

**Addressing / IDs / Encoding.**

Events / Messaging.

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

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.

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

**Dataflow: Events. Reactive APIs.**

Addressing. Reactive (Events, Dataflow). Graph encoded behavior (encoding / patterns). Reactive objects (Model, Layer / Statement, Resource, URI). Dispatch: Bus / DIDs resolution.

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

Model

Message

Interaction

Transform (Augmentation)

Flows / Routes (Augmentation, signatures)

Addressing

IDs Encoding

Processor

Producer

Consumer

Subscriptions (from metadata)

Queues.

**Protocols:**

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.

Augmentation: Model, Context (Statement), Resource levels Message (quads) IO application, resolution, transform / declarative specification (template, input context, results). Dataflow contexts from Message levels application.

Augmentation: For example, a template Statement (Statement used as transform specification) from, for example, the Interaction Model, may state matching pattetns such as:

(ContextClass : Subject, Context, Occurrence, Attribute);

and, when applied to an input Message:

(Statement, Subject, Predicate, Value);

reacts emitting the following Statement, transforming input context Message according template rules (input Subject -> output Attribute):

(TransformClass : Entity, Statement, Subject, Predicate);

which is materialized in the corresponding Model and is itself again a Message routed for further processing. TransformClass is an instance / subclass of super / meta class ContextClass (model layers transform rules).

Augmentation contexts / templates: Model, Layer, Resource. Template Meta Resource(s) (Context, Occurrence, Attribute, Value, CSPO, Kind, etc.): matches context input Message Resource by context extending / implementing / instantiating such Meta Resource(s).

Transforms: explicit template resources / model layer resources as input / specification (i.e.: apply a Role to a Class from Source Model: Entities playing such Role as results). Model Resource as template outputs common supertypes with context input as Message result.

Augmentation. Dialog. Query API.

Forms. Templates.

Ontology levels / layers.

Augment / Activate Resource (via addressing).

**Protocols:**

Hypermedia addressing and annotations. Extended content types annotations: request accept: image/png;people, response content type: text/xml;facesCoords.

Addressing: according content type (i.e.: response XML dialect for coordinates in an image / hash determining anchor in an HTML document) renders corresponding object (DOM document in this case) for “activation” on addressed parts.

Context signatures.

Activation (parse gestures / render content according context). Browser.

URIs scheme. Extended Content type. Message dialog (peers Augmentation).

**Protocols:**

Goal, Purpose: Fulfill Context.

Forms / Templates.

Dialogs: Model I/O (Message) flows.

**Models browsing / discovery APIs.**

HAL / OData like.

**Services (URIs APIs)**

Index

Naming

Registry

Service (URIs APIs). Index. Naming. Registry. Custom (signatures : Predicate Kind).

**Data / Reference Model.**

Functional declarative Semantics Specification.

**Ontology matching. Ontology levels.**

**Platform:**

Implementation (Protocols). Core, RX, Dataflow. Model: Reactive Dataflow.

(Resource : URI) : DID : Class / ID aligned Resource URIs.

DIDs encode Resource contents (hash / tensor / Context Kind) signatures. Resolution. Endpoints (provenance / contexts).

Resource: Reactive entity (Processor). DIDs: Resource Bus addresses. Container: services / nodes (models).

Bus / reactive dataflow layer (physical distributed Resource(s) events dispatch: services / nodes containers). Publish / consume Resource streams.

DID encoded Resource hash: events signatures.

Resource produced events (by Context).

Resource consumed events (by Context).

**To Do / TBD**:

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

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

Encoding. Endpoints. Dataflow.

Augmentation: common super type inference: Aggregation, Alignment, Activation. Verbs / Activation. Functors (context: messages, reified mappings: templates).

Message: specification / transform (input / output dialog domain / range). Context Kind.

Augmentation: Aggregation (Context template).

Augmentation: Alignment (Attribute, Value template).

Augmentation: Activation (Kind type inference, Class / ID resolution / alignment: semiotic / encoding templates).

Augmentation templates: Metagraph.

Core Backend APIs.

Node Quad Store Backend. Sync DIDs.

RDF / OWL Backend URIs (Statement Context / Resource addresses, services).

DIDs: decentralized persistence. Event sourcing. Sync Backend. Identifiers for (reified) meta Resource (URI, Resource, Statement, Context, Kind).

Protocol / Dialog: I/O. Prompts.

Application Ontology Levels:

Backend

Session

Frontend / Service

Domain Ontology Levels (DCI layers). Application ontology Aligned.

Ontology levels: data / schema / behavior (backend, business, frontend) objects.

Microformat like frontend / services (rendering layer) elements annotations protocol (ontology levels / contexts vars: referer, data values: price, schema rels: master detail, behavior: account transfer) for hypermedia activation rendering layer. Annotations: addressable / addresses in rendering context.

Render Wiki like abstract representations for hypermedia rendering / activation.

XML abstract representation of reactive content / behavior declarative description. Extended content types. XLink, XPointer, XQuery.

JSON / XML / XSL: XUL / ZUL / HTML (rendering frontend / services layer formats). XSLT / XPath / XLink / XPointer / XQuery.

API: URI, Resource, Message, Statement, Kind, Layers. Representation: XML bindings.

Kind : Statement : Message : Resource : URI;

URI / Resource<T extends URI> : Monad.

Resource: (URI, URI, URI, URI); URI : Resource.

Message: specification / transform (input / output dialog domain / range). Context Kind.

Resource XML Encoding (nested layers quads). Message XML Encoding.

XSLT templates (Resolution, Activation, Alignment, Aggregation). Resolution algorithm: TBD (ontology matching).

Events: Dataflow. Reactive Model endpoint Message dispatch / resolution (Producer). Resolve (addressable) Message resources (Resolution template). Apply templates (Resolved resources : model / Message resources : view context) : XML (Message).

Metagraph / Grammar (sample):

(Kind, SuperKind, Attribute, Value);

(Occurrence, Kind, SuperKind, Attribute);

(Context, Occurrence, Kind, SuperKind); (attributes / links bindings).

(Resource, Context, Occurrence, Kind); State Resource Kind in occurrence context (context / role bindings).

(Statement, Resource, Context, Occurrence); State Resource URIs occurrences / Resource class IDs (classification bindings).

(Interaction, Statement, Resource, Context);

(Action, Interaction, Statement, Resource);

Interaction / Model?

Action / Schema?

Metagraph Resource(s): class / instance IDs of reified meta Resource(s) in contexts / roles with attributes / values. Describes Model(s) : Interaction Model (Source, Dimensional, Grammar).

Resource: reactive entity. Augmentation: apply Interaction Model / input Message to parsed Resource. Reaction: matching Resource set (resolution depending Resource type).

Message: Resource aggregation (occurrence, context, model) dataflow (Augmentation). Resolves Resource Set specification.

From Intetaction Model Augmentation (patterns: CRUD / IO, Aggregation, Alignment, Activation): Source, Grammar, Metagraph, Dimensional models. TBD: Parser (consumes Resource inputs, apply Message rules, emits Resource set).

Grammar (kinds), Metagraph (contexts, meta Resource roles): Contextual / Functional Type Object (Dynamic Object Model), Actor / Role pattern models.

Kind in context: URI / Resource<T extends URI> Monad (Type Object).

Role in context: URI / Resource<T extends URI> Monad (Actor / Role).

Context: CSPO Occurrence. Actor role meta Resource.

Types / Roles: Reified Kinds / meta Resource(s).

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.

* 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.
* 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) 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.
* 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.
* Model Contexts: Meta Resources / Contexts hierarchies. Models:
* Data: Source / Interaction, Schema: Encoding / Grammar, Behavior: Dimensional / Measures (marriage).
* (Model, Behavior, Flow, Class); Model aggregation layer.
* Ontology Matching. Semiotic. Sets. Functional Reference Model.
* (Context, Sign, Concept, Object);
* Dimensional alignment / aggregation layers (lower resource alignment layers):
* (Value, Distance, Prev, Next : in Units); (Measure, Value...) (Unit, Measure, Value,...); (Resource, Unit, Measure, Value);