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

Trust. Consistency. Event sourcing. Inferencing (of distributed state). Reconciliation.

Certify Entity / Subject Identity. Class / instance alignment (matching).

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

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

Objectives:

Develop Protocol (APIs) to facilitate Enterprise Application Integration (EAI) by means of Semantic technologies and Machine Learning. Ontology matching driven data, schema, behavior inference / aggregation / matching. Reasoning and learning over different consolidated backends alignments.  
  
Distributed P2P (Blockchain) approach of data synchronization between peers for ease of deployment patterns election and datasources integration (APIs, microservices, etc.).  
  
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, equivalent attributes, equivalent roles).  
  
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).

Augmentations:  
  
Activation (type inference): classification (determine class / metaclass / roles for entity attributes and values).  
  
Alignment (infer attributes / relations): clustering (from multiple occurrences of same entity in diverse data sources).  
  
Aggregation: infer roles in contexts: regression (Person class in Employment interaction : Developer role).  
  
Integration of addressable resources. Reactive I/O (sync back ends). Content type driven semantic augmentation / annotations.

Augmentation of distributed resources. Annotations (Semantic / ML). API for resource / schema / interactions exploration / protocol for message based API "dialogs" execution. HAL (Hypertext Application Language), OData (REST) like interfaces.

Hypermedia Activation. Addressing. Link extended content types resources elements / parts with other resources addressed elements.

Domains: data, schema and behavior of business applications (ERP, CRM, BI, SCM, HMS, etc.).

General purpose business domains problem resolution / tasks, goals accomplishment helper tools.

Syndication (contextual hypermedia activation): QA. Polls. Learning. Profiles. Guided task (wizards), guided editors: Context: Goal / Purpose.

First, I'll try to describe a "problem" (problem "spaces" in this case) and how a Purpose driven user Community achieves its Goal(s) by means of Goods, Products and Needs satisfaction (ontology levels: from abstract upper ontology to user gesture command in user interface / service invocation).  
  
The problem is to organize interdisciplinary (multiple domains) Task(s) in a Purpose fulfilment network with Actors, Contexts and Roles (with attributes and values). Problem spaces (domains) are declaratively stated by DCI[1] design pattern: Data / Context / Interaction use cases definitions and instances.  
  
Collaborative Federated Actor network complying determinate Profile(s) satisfying specific Product / Good / Need abstraction playing determinate Role in use cases Context.  
  
Domain Translation between business domains, example: orders, delivery, invoicing (micro) services Model instances are the means by which distributed disparate data, schema and behavior of different sources (applications, services) integration could be performed by means of Semantic Intelligence and Augmentation Protocol(s).  
  
A domain can be defined in terms of a set of actions / tasks with the Purpose of satisfying some Goal solving the Need for a Good producing / gathering a Product. Ontology. Purpose as Goal “class”.

The principal focus is to deploy a (social) Collaborative peer (Actor) network for which entities and individuals develop Profile(s) which acquaint them with Purpose resolution capabilities. Then, according peer’s specific needs (domain Goals) the application orchestrates interactions needed for Product(s) Task(s) accomplishment.

URIs API for annotating network retrieveable resources metadata. Content type / model driven augmentations / activations (models features / outputs). Subject attributes / values. Occurrences contexts / roles. Paths, pointers, locators. Example: annotate document URIs (parts, sections, mentions), annotate images URI (whole image description, coords: classes, individuals), annotate DB, table, row, column, value URIs, annotate / describe service / APIs URIs. Hypermedia protocol composable with other (described / annotated) APIs / resources. Example: Drive APIs.

What my attempts are about where, in the beginning, to match different URIs or 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.  
  
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.  
  
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.  
  
Layers shape is as follow:  
Resource : Functional URI wrapper.  
(Context : Resource, Occurrence : Resource, Attribute : Resource, Value : Resource);  
  
Each layer abstract:  
  
Statement (data instance):   
(Statement, Occurrence, Attribute, Value);  
someOne buys someProduct  
  
Entity (data class):  
(Entity, Statement, Occurrence, Attribute);  
someBuyer, someProduct (Entity);  
  
Role (schema instance):  
(Role, Entity, Statement, Occurrence);  
Buyer, Product (Role);  
  
Class (schema class):  
(Class, Role, Entity, Statement);  
Person, Good (Class);  
  
Flow (behavior instance):  
(Flow, Class, Role, Entity);  
someBought (Flow);  
  
Behavior (behavior class):  
(Behavior, Flow, Class, Role);  
Buy (Behavior);  
  
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 metamodel. The act of applying an Augmentation implies one source Resource (context), one template Resource (transform) and a resulting (set of) Resource(s).  
  
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.

Application:

features / techniques / patterns.

Implementation deployment use cases. Sample Apps: SoLiD / PIM / PASCEN: App declaratively built with framework, Implementation Integrations.

Extension / Augmentation: BI / EAI. Smart dashboards / reports / workflow / process / activity components. Activable smart indicators / components (predict / execute). Declarative Model interpretation into abstract application models. Rendering (Gestures ontology).

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

Serialization. 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: 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.

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

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).

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

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).

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

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.

**Services (URIs APIs)**

Index

Naming

Registry

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

**Data / Reference Model.**

Functional declarative Semantics Specification. Semiotic / Dimensional alignment layers. TBD.

**Ontology matching. Ontology levels.**

Semiotic / Dimensional alignment. TBD.

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.

**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: Dataflow by Context Kind domain (Subject Kind) / range (Object Kind).

Ontology Matching. Upper ontologies. Primitives.

**Addressing / IDs / Encoding.**

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:**

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.

**Models:**

Models: Meta Model / Resources. Model source / grammars / interactions. Upper semiotic / dimensional layers.

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

Explain layers, Meta Resource(s), Context (class / instance / metaclass) / Kind hierarchies. Augmentation behaviors description.

Meta Model (Meta Resources)

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

Source Model. Data.

Grammar Model. Schema.

Interaction Model: Behavior?

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).

Model Contexts: Meta Model Meta Resources reified Contexts hierarchies. Models:

Data: Source / Interaction, Schema: Encoding / Grammar, Behavior: Dimensional / Measures.

(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); Marriage example.

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);

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

**Interaction (Meta) Model Specification.**

Aggregation (data)

Alignment (schema)

Activation (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);

Data: 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).

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

Behavior: 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);

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?

**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.

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.

Events metamodel (TBD):

(Object, State, Axis, Type)  
(State, Axis, Type, Event)  
(Axis, Type, Event, Event)  
(Type, Event, Event, Event)  
(Event, Event, Event, Event)

**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);

**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 (core Meta Model):

Data (Aggregation);

Schema (Alignment);

Behavior (Activation);

Interaction (Meta) 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: metamodel / custom (domain).

Message - Model - Template - Augmentation - Transform - Model - Message.

Encodings: Models

Functional / Signature IDs.

Grammars.

Message: Resource.

Model event. Data.

Resource ID / Set specification.

Model: RDF. Resource layers.

Reified Models. Upper (Semiotic / Dimensional) layers Alignment. Ontology Matching.

Template: Resource. Grammar.

Model state. Context.

Functors.

Augmentation:

Model I/O, Dialog. Interaction.

Algorithm: parsing, declarative.

Transform: Resource.

Results (dataflows).

Materialize.

Model

Message

Ontology / Persistence.

Functional Reference Model.

CRUD (events).

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.

Resolve Message matching Resource from behavior layers / matching kinds from Model / data layers.

(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);

Example: a message composed of a kinds CSPO matches statements “instances” of those specifications (statements whose CSPO have matching kinds). A message with three CSP kinds and a (potentially unknown) object URI retrieves matching resources having that object value into corresponding property kinds. An statement of plain (potentially unknown) URIs instantiates / updates and augments new / known resources added to models and returns an augmentation transform result.

Interaction Model: Context of Messages model for a given interactions session / dialog state. Message invocation requests: Statement(s) building Resource invocation graph with layers matching Message patterns. Layers graph invocation patterns matching from higher to lower layers resources fulfilling higher layers templates. Variables, wildcards, placeholders.

Dialog arguments resolutions example: higher layer Resource / Message request / invocation instantiates in Interaction Transform context corresponding lower layer graph statements to be “populated” to fulfill request. Message IO of “forms” (Messages) inter-peers (originating peer  
acting as “server”) for initial requested peer to “ask” for form elements to be populated (interaction context “dialogs”). Resolution may propagate to other peers (content aware addressing dataflow routes dispatch: P2P resources address encodings, matching forms models requests). Nested interactions.

Explain messages (resource resolution). Grammar. Match model Resource(s). Compound nested CSPO statement contexts defines result behaviors. Message CSPO contexts may define create, retrieve, update or delete operations (passing 'null' for example for resource / statement to be deleted).

Explain transforms (message application). Transform: Resource stream result of Message application over resolved Resource(s)). Input statements: Message(s) / Resource(s) (from input message or to be populated or populated in dialog) and "goal" Message / Resource aggregating a model from Resource MetaGraph with Message / Resource bindings.

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.

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.

**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).

**Model I/O Dataflow:**

Dataflow: Events. Reactive APIs.

Augmentation: basic operation.

Events declarative definition. State change of value in axis in measure of context.

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).

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.

Resource: Reactive entity (events source / sink) wrapping an URI endpoint implementing some kind of I/O, Signature: Resource Context Kind. Matching “ranges” (SK) dispatch matching events to matching “domains”.

DIDs: Encoding (signature / contents) identifier. Endpoints: provenance. Address: Messaging bus. Discover signatures, contents, potential transform results.

Dataflow:

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

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 (Deployment / use cases):**

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).

Extension / Augmentation: BI / EAI. Smart dashboards / reports / workflow / process / activity components. Activable smart indicators / components (predict / execute). Declarative Model interpretation into abstract application models. Rendering (Gestures ontology).

**Protocols (Deployment / use cases):**

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. Signatures activation (JAF) interactive dashboards.

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

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

Goal, Purpose: Fulfill Context.

Forms / Templates.

Dialogs: Model I/O (Message) flows.

Models browsing / discovery APIs.

HAL / OData like.

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).

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.

Application augmentations / extensions (connectors):

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.

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

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