* Layers, input layer: Model Roles. Aggregation: Layer Roles shifting until full Layers Roles Statements. Layer: Augments Models.
* Core Statements Roles / Shapes:
* Occurrence: (Class, Instance, Attribute, Value);
* Occurring: (Class, Instance, Occurrence,  Role);
* Statement Transforms / Relations: order, equivalence, roles, etc.
* Layer Template Mappings:
* Template : Context : Statement : Resource Layer Roles Monads.
* Layers Aggregation (rotating role types values for each layer from V to C):
* Layers Aggregation: Perform Layers Templates Matching  Augmentations. Augmentation  Models Statements Metadata producing new potentially matching Contexts.
* (Template, Context, Statement, Resource):
* Canonical Template Mapping Layers: Aggregation of Template Matching Models Layers until first layer Value is wrapped into final layer Context. Perform Layers Augmentations.
* Monads. Wraps Models Roles. Matching: Patterns wrapped Resource Roles values: placeholders, variables, wildcards, reified instances.
* Core Roles (Wrappers):
* Resource : Monad(x : Resource);
* Statement : Monad(x : Resource) : Resource, Quad DTO Statement.
* Context : Monad(x : Resource) : Statement;
* Template : Monad(x : Resource) : Context.
* Template : Context : Statement : Resource
* Wrapper / Wrapped reactive / event driven message bus API.
* Quads I/O: Base Model Protocol. URN Encoding / Content Types. Transforms / Mappings: browse links transforms. Layers: Augments Models (resolve / merge ontologies).
* Augmentations:
* Data Match Model (Sets).
* Infer Kinds / Flows.
* Models:
* (Statement, Kind, Attribute, Value);
* (Template, Statement, Kind, Attribute);
* (Template, Context, Statement, Resource);
* Schema Match Model (Graph).
* Infer Types / Hierarchies / Shapes.
* (Statement, Resource, Kind, Attribute);
* (Template, Statement, Resource, Kind);
* (Template, Context, Statement, Resource);
* Roles / Behavior Match Model. (Roles). Infer Contexts / Interactions / Measures.
* (Measure, Dimension, Unit, Value);
* (Resource, Measure, Dimension, Unit);
* (Statement, Resource, Resource, Unit);
* (Template, Context, Statement, Resource);
* (Roles with original wrapped Values).
* Layers: Models Aggregation. Layer Contexts: upper previous layer Resource value wrapped in Context Resource Monad.
* (Amantes, Pedro, Ama, María);
* (MariaAmadaPor, Amantes, Pedro, Ama);
* (Amar, MariaAmadaPor, Amantes, Pedro);
* (PedroAmaA, Amar, MariaAmadaPor, Amantes);
* (Amor, PedroAmaA, Amar, MariaAmadaPor);
* Inputs: Aggregate SPO into CSPO: Aggregates Contexts Type / Table / Class Kinds. Aggregate PK Cols, Cols : Occurrence, Val : Resources.
* Inputs (Rel / Graph): (Type / Table / Class, PK : Resource, Col : Occurrence, Val : Resource).
* Inputs (Rel / Graph) FKs: Val : Resource equivalent PKs.
* Augmentations / Transforms:
* Data matching. Resource equivalence: identity / comparisons / order transforms.
* Schema matching. Predicates equivalence. Domain / Range types. Order: data flow contexts.
* Behavior matching. Domain / Range values applied functional predicates identity. Order: data flow interactions.
* Transforms: Resource Roles browsing, i.e.: [rsrc.occ.role.rsrcs.ctx.roles](http://rsrc.occ.role.rsrcs.ctx.roles). Resource Roles getters.
* Data layer: matching alignment transforms. [rsrc.type.rsrcs](http://rsrc.type.rsrcs). Resource / Services augmentations.
* Encodings: Ternary. FCA. Graphs. Primes / Bitstrings. Tensors.
* Layers Augmentation. Data, Schema, Behavior Layer wise Statement Roles.
* Alignment: Data Matching Augmentation.
* Activation: Schema Matching Augmentation.
* Aggregation: Behavior Matching Augmentation.
* Core Statements Roles:
* Occurrence: (Class, Instance, Attribute, Value);
* Occurring: (Class, Instance, Occurrence,  Role);
* Layer Template Mapping:
* Template : Context : Statement : Resource Layer Roles Monads.
* Layers (rotating role types values for each layer from V to C):
* (Template, Context, Statement, Resource):
* Canonical Template Mapping Layers: Aggregation of Template Matching Models Layers until first layer Value is wrapped into final layer Context. Perform Layers Augmentations.
* Monads. Wraps Models Roles. Matching: Patterns wrapped Resource Roles values: placeholders, variables, wildcards, reified instances.
* Core Roles (Wrappers):
* Resource : Monad(x : Resource);
* Statement : Monad(x : Statement) : Resource, Quad DTO.
* Context : Monad(x : Context) : Statement;
* Template : Monad(x : Template) : Context.
* Wrapper / Wrapped reactive / event driven message bus API.
* Quads I/O: Base Model Protocol. URN Encoding / Content Types. Transforms / Mappings: browse links transforms. Layers: Augments Models (resolve / merge ontologies).
* Augmentations:
* Data Match Model (Sets).
* Infer Kinds / Flows.
* Models:
* (Statement, Kind, Attribute, Value);
* (Template, Statement, Kind, Attribute);
* Schema Match Model (Graph).
* Infer Types / Hierarchies / Shapes.
* (Statement, Resource, Kind, Attribute);
* (Template, Statement, Resource, Kind);
* Roles / Behavior Match Model. (Roles). Infer Contexts / Interactions / Measures.
* (Measure, Dimension, Unit, Value);
* (Resource, Measure, Dimension, Unit);
* (Statement, Resource, Resource, Unit);
* (Template, Context, Statement, Resource);
* (Roles with original wrapped Values).
* Aggregate Layers Contexts: Value Context Upper Shifting Layers. Model Layers Augmentations.
* Layer Aggregation: Perform Augmentation of Aggregated Layers Data, Schema, Behavior Templates. Functional Model Data Flows CRUD.
* Aggregation: Augment Aggregated Layers Data, Schema, Behavior Templates. Functional Model Data Flows CRUD.
* Template : Statement : Resource Monads. Matching: Augmentable Resources Inputs Augmentation: Data, Schema, Behavior activation, agreggation, alignment into Layers browse / expansion.
* Template : Statement : Resource Monads. Inputs Augmentation: Data, Schema, Behavior activation, agreggation, alignment.
* Monads. Wraps Models Roles. Matching: Patterns wrapped Resource Roles: placeholders, variables, wildcards reified instances.
* Resource : Monad(x : Resource);
* Statement : Monad(x : Statement) : Resource, Quad DTO.
* Template : Monad(x : Template) : Statement.
* Quads I/O: Base Model Protocol. URN Encoding / Content Types. Transforms / Mappings browse links transforms.
* Models:
* Augmentations:
* Sets. Data Match Model.
* Resource Context: Data Activation, Kinds Attribute Values matches layer Resources. Ontology matching.
* Functional Expansion Transform Mappings: Augmentations domain transforms, Resource Roles APIs.
* Match: (Template, Resource, Statement, Kind);
* Functional Expansion Transform Mappings: (Resource, Statement, Kind, Value);
* Graph. Schema Match Model.
* Statement Context: Schema Aggregation, Resource Kinds Attributes matches layer Statements.
* Match: (Template, Statement, Resource, Kind);
* Functional Expansion Transform Mappings (Statement, Resource, Occurrence : Kind, Attribute);
* Behavior: Reified Mappings / Transforms / Functions:
* Roles. / Dimensional / Discrete. Behavior Match Model. Match: (Template, Dimension, Measure, Unit : Kind); Aggregate Contexts and Interactions.
* Match: (Template, Dimension, Measure, Unit : Kind);
* Functional Expansion Transform Mappings: (Dimension, Measure, Unit, Value);
* Template Context: Behavior Alignment, Template Statement Resource Kind matches layer Template Roles.
* (Template, Statement, Resource, Kind);
* Layers. Apply Templates. Templates Matching populates augmentatiom layers: Data, Schema, Behavior matching Templates and applying Transforms: Models schema signatures (roles).
* API: Functional DOM Reactive / REST HATEOAS. Browse Monads Roles Mappings. Subscriptions: Materialized Template dataflow context bindings: Patterns.
* Template : Statement : Resource Monads. Matching: Augmentable Resources Inputs Augmentation: Data, Schema, Behavior activation, agreggation, alignment Layers.
* Monads. Wraps Models Roles. Matching: Patterns wrapped Resource Roles: placeholders, variables, wildcards.
* Quads I/O: Base Model Protocol. URN Encoding / Content Types. Transforms / Mappings browse links transforms.
* Models:
* Sets. Data.
* Graph. Schema.
* Roles. / Dimensional / Discrete. Behavior.
* Layers. Apply Templates
* Augmentations:
* Resource Context: Data Activation, Kinds Attribute Values matches layer Resources. Ontology matching.
* (Resource, Occurrence : Kind, Attribute, Value);
* Statement Context: Schema Aggregation, Resource Kinds Attributes matches layer Statements.
* (Statement, Resource, Occurrence : Kind, Attribute);
* Template Context: Behavior Alignment, Statement, Resource, Kind matches layer Templates.
* (Template, Statement, Resource, Kind);
* API: Functional DOM Reactive / REST HATEOAS. Browse Monads Roles Mappings. Subscriptions: materialized Template dataflow context bindings: Patterns.
* Resource Context: Data
* (Resource, Occurrence, Attribute, Value);
* Statement Context: Schema
* (Statement, Resource, Attribute, Value);
* Template Context: Behavior
* (Template, Statement, Resource, Attribute);
* Template : Statement : Resource.
* Reified Mappings / Transform. Template Patterns Roles populates corresponding Quad item with matching source Resource
* Models. Input Quads. Monad / Roles DTOs. Graph. Templates / Mappings.
* Augmentations: via Templates / Mappings.
* Activation: Sets, type inference. Data Matching Templates.
* Aggregation: Roles. Layers. Schema Matching Templates.
* Alignment: Discrete / Dimensional frontend aggregation / expansion APIs. Behavior Matching Templates.
* Shapes: "uncle"
* Resource : Monad(Resource);,
* Templates: Augmentations. Aggregate, Match / Activate, Expand / Align.
* Templates: reify Transforms. Aggregate Template Statements: mappings Resource value domains, predicates, ranges. Matching / Dataflow.
* Template Statement: Statements aggregates Kinds Occurrences for Context Resources.
* Template : Statement : Resource.
* Resource: Matching Mapping Statement Contexts. Resource Roles Mappings. Resolve Resource Role Monad for each Transform Quad DTO CSPO. Matching: Resource declarative Mappings aggregation.
* Statement: Apply Matching Contexts Mappings. Transform input Statements into CSPO / Graphs Role Alignments Mappings.
* Template: Reified Mappings. Aggregate matching input Statements. Graph / Models / Augmentations declarative (Mappings) Transform statements.
* Template Match: Patterns. Aggregation. Reified Roles Templates populates source Statements corresponding Role values mappings. Multiple Patterns Template aggregated as Statements Resources.
* (Template, SK, PK, OK) : Template;
* Template: Context Resource Monad expression (reified Roles). Aggregate Mappings.
* Multiple (aggregated) Template Statements renders different Statements to be merged for different Resources.
* Statement Apply: Render Template Pattern Statements given matching input Statements.
* Resource Resolution: Template / Statement matching. Aggregated Templates merge.
* Aggregated Template Statements. Populate from upper ontology inferred from inputs / Mappings invocation.
* (SK, PK): Possible Mappings.
* (PK, Empty Object): New Mapping.
* Statement : Monad(Statement);
* Statement : Resource.
* Statement: CSPO / Templates Mappings Transforms.
* Statement: Monad of Resource Quad (DTO).
* Statement Values : Context aggregated CSPOs.
* Statement as Resource:
* (C, (CSPO), P, O);
* (Amar, (Empleo, Pedro, Role, Manager), Ama, María);
* Matching / Aggregation: reified statements, kinds. roles, variables, wildcards.
* Matching: Patterns. Template Kinds matches Statements Kinds. Activation. Reified Statements Shapes.
* Matching Alignment. Apply Expansion: CSPO Transforms Statements results.
* Resources: Statements, Model Layers CSPO Roles. Resource Monad types / values. Uniform transforms API between models roles augmentations browse. (SAILs).
* Models:
* Sets: (Statement, Kind, Attribute, Value);
* Graph: (Context / Type, Resource, Attribute, Value).
* Graph: (Context : Type, Subject : Node, Attribute : Arc, Value : Node);
* Graph: (Resource, Context / Type, Attribute, Value).
* Roles: (Class, Instance, Occurrence, Role);
* Roles: (Metaclass, Role, Context, Instance);
* Discrete Roles: (Relationship : Metaclass, Relation : Role, Kind : Context, Resource : Instance);
* Dimensional Roles: (Dimension : Class, Measure : Instance, Kind : Occurrence, Value : Role);
* MVC / DCI: (Resource, Data, Context : Form, Interaction : Behavior); Layer Templates. Augmentations.
* Augmentations:
* Template Statements: Matching (Data), Aggregation (Layers), Expansion (Alignment).
* Matching: matching reified role classes hierarchy. Reified Relations: common superclass, order, others. Comparisons / roles.
* Data Activation:
* Augmentation: Sets Activation. Populate Models and their aggregated SPO inputs Statements. Aggregate SPO Kinds: type inference and matching. Render (Type, Resource, Attribute, Value) Statements. Type: Reified Kind (SPO), Attribute / Value of complementary SPO of Resource. Render (Resource, Type, Statement) Statements. Statement: Reified Statement in corresponding set. Data Alignment. Sets / Graph Models.
* Schema: Layers Aggregation. Layers Templates matches previous Layer Value with current Layer Context.
* Augmentation: Layers Aggregation. Populate Layers and their aggregated schema values context types. Different Models Aggregations yields equivalent Layers Statements shapes arrangements of Aggregations Models Schema roles. Layers Aggregation. Graph / Roles Models.
* Behavior: Layers Alignment.
* Augmentation: Behavior: Alignment. Entail aggregation Layers Statements expanded CSPOs. Entail Data Flow. Functional Data Flow Contexts. Expanded (possible / matchings) Statements traversal. Behavior Alignment. Discrete / Dimensional Models.
* Layers: Models Aggregation. Layer Contexts: upper previous layer Resource value wrapped in Context Resource Monad.
* (Amantes, Pedro, Ama, María);
* (MariaAmadaPor, Amantes, Pedro, Ama);
* (Amar, MariaAmadaPor, Amantes, Pedro);
* (PedroAmaA, Amar, MariaAmadaPor, Amantes);
* (Amor, PedroAmaA, Amar, MariaAmadaPor);
* Inputs: Aggregate SPO into CSPO: Aggregates Contexts Type / Table / Class Kinds. Aggregate PK Cols, Cols : Occurrence, Val : Resources.
* Inputs (Rel / Graph): (Type / Table / Class, PK : Resource, Col : Occurrence, Val : Resource).
* Inputs (Rel / Graph) FKs: Val : Resource equivalent PKs.
* Augmentations / Transforms:
* Data matching. Resource equivalence: identity / comparisons / order transforms.
* Schema matching. Predicates equivalence. Domain / Range types. Order: data flow contexts.
* Behavior matching. Domain / Range values applied functional predicates identity. Order: data flow interactions.
* Transforms: Resource Roles browsing, i.e.: [rsrc.occ.role.rsrcs.ctx.roles](http://rsrc.occ.role.rsrcs.ctx.roles). Resource Roles getters.
* Data layer: matching alignment transforms. [rsrc.type.rsrcs](http://rsrc.type.rsrcs). Resource / Services augmentations.
* Encodings: Ternary. FCA. Graphs. Primes / Bitstrings. Tensors.
* Functional Facades:
* DOM. Data Flow. Functional DOM REST HATEOAS Facades. Browse / Functors.
* Contexts. Transforms.
* Upper / Matching (Business)
* Units of Measurement (continuos) APIs /  Ontology.
* Discrete (events) APIs / Ontology.
* CDI: Signatures Functional Dataflow Injection / Bus Resolution.
* Aggregation Statements Types:,
* (Kinds / Types, Object / Resource, Predicate / Arc, Object / Resource);
* (Object / Resource, Kinds / Types, Predicate / Arc, Object / Resource);
* Encoding / Matching:
* Functional Primitives: (Matching). Graph Shapes Model. Layers. Example: reify / render / match "uncle" relation / "marriage" situation from graph statements.
* Layers Augmentation. Data, Schema, Behavior Layer wise Statement Roles.
* Alignment: Data Matching Augmentation.
* Activation: Schema Matching Augmentation.
* Aggregation: Behavior Matching Augmentation.
* Canonical Statement:
* (C, S, P, O);
* Canonical Type Statement
* (Type, Resource, Attribute, Value);
* Canonical Subject Statement
* (Resource, Type, Attribute, Value);
* Type centric Statement (Dimensional)
* Subject centric Statement (Discrete)
* Reification: Meta Models, Statement Layers Roles
* Layers: Data, Schema, Behavior
* Upper Hierarchies of Metaclass, Class, Instances, Contexts, Roles, Occurrences in Layers Contexts Statements. Meta Model Primitives: Layers Templates.
* Encodings
* Functional API Mappings: Monads / Transforms Model / Domains Dataflows.
* Layer Templates
* Declarative HATEOAS Endpoint
* Functional DOM Client / Server Facade
* Local Client Features Facades. Plugins. Generic REST. ESB Message Templates. Adapters. Local Services Facade (RDBMS, REST, Soap, WSDL, etc.) emulating augmented original sources
* Augmentation Layers:
* Type centric Statements (Dimensional)
* (T: Dimension, D: Measure, P: Unit, D: Value);
* Subject centric Statement (Discrete)
* (D: Relation, T: Relationship, P: Predicate, D: Value);
* Upper Kinds Hierarchies of Metaclass, Class, Instances, Contexts, Roles, Occurrences Meta Model Primitives in Layers Contexts Statements.
* Data, Schema, Behavior Layers Augmentation Statements shifts Quads Statements Resources from more specific to more upper and general ontology Contexts Roles (primitives). Aggregation.
* Upper Ontologies: From Primitives to Form Gestures.
* Canonical CSPO Statements Reification / Rendering (aggregation / expansion) of Type / Subject Statements.
* Layers Augmentations. Data, Schema, Behavior Statement Layers Meta Model Primitives Roles (Types) Shifting on each Layer Aggregation.
* Model Augmentations / Domain Augmentations. Transforms. Templates. Matching Data Instances (Functor / Mapping) domain / range types / values dataflow data / schema / behavior.
* Templates Statements basic application: between two Statements: given a Template Statement relation URN (i.e.: common superclass, common occurrence context, etc.) Aggregates Aligned Activated Schema results. Nested URNs (context).
* Encoding URN to / from Statements: Dataflow Encoding. Infer Possible Model / Domain Functors / Transforms. Statement Layers, Meta Model Roles: routes / paths (recurse) Zippers. Resolve IDs / URNs Mappings.
* Relation Statements: Match Resources / Reified Roles Templates.
* Reify Layers (Data, Schema, Behavior) Positional Meta Model Roles (Metaclass, Class / Instance, Context / Role, Occurrence) as Predicates / Attributes. Resource Values. Template Matching.
* Reify Statements, Kinds, Resources. Templates Matching.
* Reify CSPO.
* Reify Context, Node, Concept, Node. (Statement Types).
* Upper / Matching (Business)
* Units of Measurement (continuos) APIs / Ontology.
* Discrete (events) APIs / Ontology.
* SAIL Layers (Storage and Inference Layers):
* Ternary Backend SAIL
* Sets Backend SAIL
* Statements / Quads Backend SAIL
* Services / Layers Quads Protocol.
* Layers Backend Components (Quads I/O) Services:
* Quads: Services / Layers Protocol
* CDI: Signatures Functional Dataflow Injection / Bus Resolution.
* Aggregation Statements Types:,
* (Kinds / Types, Object / Resource, Predicate / Arc, Object / Resource);
* (Object / Resource, Kinds / Types, Predicate / Arc, Object / Resource);
* Aggregation Statements Types:
* (Kinds / Types, Object / Resource / Predicate / Arc, Object / Resource);
* (Subject / Context, Kinds / Types, Predicate / Arc, Object / Resource);
* Reification. Infer Kinds / Types vía repeated matching of Statements Attributes occurrences. Quads Encoded. Attribute relation: Mapping from and edge of whim an Arc / Predicate is source.
* Encoding: Reification of Statement types to canonical form:
* (Context, Subject, Attribute, Value);
* Aggregate / de Aggregate mappings expansions.
* Operations: Monads, Transforms, Dataflow. Templates / Mappings for processing representations. TODO.
* Data Matching Services (Nodes / Resources)
* MVC: Model; Index Service
* DCI: Data; Index Service;
* Schema Matching Services (Kinds / Types)
* MVC: View (flows / prompts); Registry Service;
* DCI: Context; Registry Service;
* Behavior Matching Services (Predicates / Arcs)
* MVC: Controller; Naming Service;
* DCI: Interaction; Naming Service;
* Augmentations:
* Alignment (Data Matching)
* Activation (Schema Matching)
* Aggregation (Behavior Matching)
* Index Service
* Naming Service
* Registry Service
* DDD (Meta Circular Interpreter) : Code as Data.
* Prompts / Flows: View Forms.
* Meta Model : Dataflow
* Use Cases
* Kind / Type Sets / Quads: Resource : Statement
* Kind / Type
* Statement:
* (Context : Statement, Subject : Resource, Attribute : Kind, Value : Resource);
* Value Resource according Kind (Roles) Statement Subject occurrence.
* Sets: de aggregated Statements. Reification. Layers (Models) expands Statements for each Kind Attribute.
* Graph:
* (Context : Type, Subject : Node, Attribute : Arc, Value : Node);
* Kind / Type: Aggregated Attribute / Arc occurrences. Type Inference.
* Type Context Arcs Expansion. Augmentations.
* Translation Layers:
* Ternary Translation Layers:
* Input: Parse Quads : Emits Graph Statements Roles.
* Augmentations
* Output: Visit Ternary Mappings. Emit Statements.
* Sets Translation Layers:
* Input: Parse Quads : Emits Graph Statements Roles.
* Augmentations
* Output: Visit Ternary Mappings. Emit Statements.
* Quads Translation Layers:
* Input: Parse Quads : Emits Graph Statements Roles.
* Augmentations
* Output: Visit Ternary Mappings. Emit Statements.
* Ternary Translation Layers:
* Augmentations / Matching.
* Ternary: (Sign, Concept, Value);
* Resource Roles:Ontology Matching: Wrapper / Wrapped Types.
* Data, Schema, Behavior Monads / Functors.
* Object, Concept, Value Roles.
* (object) : (type) : (arc)Arc : Type object occurrences.
* Property Graph Nodes / Arcs Quads Augmentation I/O: Rendering / Parse. SAIL Backend specific encoding / transform.
* Encoding / Matching:
* Functional Primitives: (Matching). Graph Shapes Model. Layers. Example: reify / render / match "uncle" relation / "marriage" situation from graph statements.
* Primitive slots template Quads (Roles) populating "shapes" matching promps / flows / state transitions.
* Templates: Relation Statements Resources. Reified Meta Model.
* Order: Functional Dataflows. Functor / Transform domain / range Activations.
* Relations: Reified Monads / Transforms Mappings (data, schema, behavior) levels. Wrapper Type, Wrapped Type, Wrapped Value, Transform domain / range. To Do.
* Metaclass
* Class
* Instance
* Context
* Role
* Occurrence
* Primitive Relations.
* Functional Transforms: Browse State / Traversal. Navigational Context (Primitives State) Builder of Mappings (Templates).
* Augmentations: Matchings.
* Abstract Functional Quads encoding / representations.
* Functional DOM REST HATEOAS Facades. Discovery.
* Upper Ontology: Need, Product, Good, Purpose
* Upper Ontologies: From Primitives to Forms / UI Gestures.
* Units of Measurement (continuos) APIs /  Ontology
* Discrete (events) APIs / Ontology
* Templates / Transforms:
* XML / XSLT Like for Semantically Aggregated Layers of Statements:
* Reify Layers (Data, Schema, Behavior) Positional Meta Model Roles (Metaclass, Class / Instance, Context / Role, Occurrence) as Predicates / Attributes. Resource Values. Templates Matching.
* Reify Statements, Kinds, Resources. Templates Matching.
* Reify CSPO. Reify Context, Node, Concept, Node. (Statement Types).
* Canonical Statements Reification / Statements Aggregation of Type / Subject Statements Types.
* Layers:
* (C, S, P, O);
* (O, C, S, P);
* (P, O, C, S);
* (S, P, O, C);
* (C, S, P, O);
* Relation Statements: Into Augmentation: Layer wise Data, Schema, Behavior "Templates" roles.
* Relation Statements: Match Resources / Reified Roles. Reified Meta Model Roles Resources. Resource "patterns", resolvable / reactive.
* Reify Relations / Predicates. Relation Mappings. Meta Model Positional Roles / Resources Matching.
* Templates / Transforms:
* XML / XSLT Like for Semantically Aggregated Layers of Statements.
* Layers Augmentation. Data, Schema, Behavior Layer wise Statement Roles.
* Alignment: Data Matching Augmentation.
* Activation: Schema Matching Augmentation.
* Aggregation: Behavior Matching Augmentation.
* Transform:
* Data: Model.
* Schema: Match. View / Context. Multiple ordered / recursive matches of Data Templates.
* Behavior: Apply. Controller / Interaction.

To Do:

* Resources: Statements Model Layers CSPO Roles Monad types / values. Uniform transforms API between models (SAILs).
* Sets: (Statement, Kind, Attribute, Value);
* Graph: (Context / Type, Resource, Attribute, Value).
* Graph: (Context : Type, Subject : Node, Attribute : Arc, Value : Node);
* Graph: (Resource, Context / Type, Attribute, Value).
* Roles: (Class, Instance, Role, Context);
* Roles: (Metaclass, Context, Role, Instance);
* Discrete Roles: (Relationship : Metaclass, Relation : Context, Kind : Role, Instance : Resource);
* Dimensional Roles: (Dimension : Class, Measure : Instance, Kind : Role, Value : Resource);
* Statements: Templates. Inputs: Aggregate SPO into CSPO: Aggregates Contexts Type / Table / Class Kinds. Aggregate PK Cols, Cols : Occurrence, Val : Resources.
* Layers: Models Aggregation. Layer Contexts upper Matching previous layer Values. Populate model upper.
* (Amantes, Pedro, Ama, María);
* (MariaAmadaPor, Amantes, Pedro, Ama);
* (Amar, MariaAmadaPor, Amantes, Pedro);
* (PedroAmaA, Amar, MariaAmadaPor, Amantes);
* (Amor, PedroAmaA, Amar, MariaAmadaPor);
* Expansion: Application. CSPOs Matching Aggregated Statements. Inputs Aggregated I/O. Matching: order / data flow.
* Aggregated SPO inputs forms Kinds Contexts CSPOs:
* Inputs: Aggregate SPO into CSPO: Aggregates Contexts Type / Table / Class Kinds. Aggregate PK Cols, Cols : Occurrence, Val : Resources.
* Inputs (Rel / Graph): (Type / Table / Class, PK : Resource, Col : Occurrence, Val : Resource).
* Inputs (Rel / Graph) FKs: Val : Resource equiv FKs.
* Graph: (Context : Type, Subject : Node, Attribute : Arc, Value : Node);
* Augmentations / Transforms:
* Data matching. Resource equivalence: identity / comparisons / order transforms.
* Schema matching. Predicates equivalence. Domain / Range types. Order: data flow contexts.
* Behavior matching. Domain / Range values applied predicates identity. Order: data flow interactions.
* (Resource, Data, Context : Form, Interaction : Behavior); Layer Templates. Augmentations.
* Resource hierarchy Monads : of(Resource). [Dimension.of](http://dimension.of)(Relationship), etc.
* Models: Sets, Graph, Roles, Dimensional, Discrete. Mappings between Model Encodings. Functional DOM traversal through encodings transforms. Models encodings represents same views of the underlying model for representation, IO, inference and functional DOM manipulation.
* Models Resources Monad: Models Getters, Resource::getModelRoles (graph, roles, dimensional, discrete CSPO roles hierachy transforms). Functional Transforms Models Alignments: Roles Resource Monads values / getters.
* Sets: Statement, Kinds, SPO Resources, Contexts. Aligns to / from aggregated Graph Statements / Contexts. Sets: (Statement, Kind, Attribute, Value); Graph: (Context / Type, Resource, Attribute, Value). Graph: (Context : Type, Subject : Node, Attribute : Arc, Value : Node); Graph: (Resource, Context / Type, Attribute, Value).
* Graph: Augmentation / IO with Sets / Roles (abstract, dimensional, discrete) aggregations.
* Statements Roles: Models Resources objects hierarchy. Getters: roles relations, Functional browsing. Services updated.
* Templates: Layers upper aggregations. Models Roles Matching: data, schema, behavior upper contexts aggregation alignments: Context shifts quad with upper aligned contexts.
* Serialization: Augmentations / Inference.
* Model: MVC / DCI Data, Schema, Behavior Common Graph Meta Model Encoding. Statements Encodings IO: Resource / Model / Services. Add SPO Statements, Services Parsing Augments Model Resource Roles, Resource Roles Population.
* Model: Aggregated Resource Roles Services Transforms Facade. Browse Augmentations Transforms: Aggregation (merge types), Alignment (merge resources), Activation (add model domain getters).
* Model Encodings:
* Common Graph Meta Model Reified Resource Roles: Metaclass, Class, Instance, Context, Role, Occurrence. Resource Superclass / Monad. Resources Attributes / Values.
* (Class, Instance, Occurrence, Role);
* (Metaclass, Role, Context, Instance);
* Service Encodings: Dimensional / Discrete (events) aggregation.
* Transforms: resources roles browsing, i.e.: [rsrc.occ.role.rsrcs.ctx.roles](http://rsrc.occ.role.rsrcs.ctx.roles). Resource Roles getters.
* Data layer: matching alignment transforms. [rsrc.type.rsrcs](http://rsrc.type.rsrcs). Resource / Services augmentations.
* Services / Augmentations (Model) I/O: MVC / DCI Encodings.
* Services: Update Model. Encodings Inferences / Facades. Dataflow, Events IO. Bus / CDI (MVC / DCI: SAILs). Transforms browse activated augmentations.
* Canonical Statement:
* (C, S, P, O);
* Canonical Type Statement
* (Type, Resource, Attribute, Value);
* Canonical Subject Statement
* (Resource, Type, Attribute, Value);
* Type centric Statements (Dimensional)
* (T: Dimension, D: Measure, P: Unit, D: Value);
* Subject centric Statement (Discrete)
* (D: Relation, T: Relationship, P: Predicate, D: Value);
* Data, Schema, Behavior Layers Meta Model Roles: (Metaclass, Class / Instance, Context / Role, Occurrence) as  Resource Occurrence / Role Attributes / Values.
* CSPO Quads Layers.
* Dimensional / Discrete:
* Type / Resource Aggregated Quad Layers.
* Sets.
* Ternary.
* FCA. Graphs. Primes / Bitstrings.
* Tensors.
* Functional Facades:
* DOM. Data Flow. Functional DOM REST HATEOAS Facades. Browse / Discovery.
* Roles. Functors.
* Contexts. Transforms.
* Augmentation Layers:
* Type centric Statements (Dimensional)
* (T: Dimension, D: Measure, P: Unit, D: Value);
* Subject centric Statement (Discrete)
* (D: Relation, T: Relationship, P: Predicate, D: Value);
* Upper Kinds Hierarchies of Metaclass, Class, Instances, Contexts, Roles, Occurrences Meta Model Primitives in Layers Contexts Statements.
* Data, Schema, Behavior Layers Augmentation Statements shifts Quads Statements Resources from more specific to more upper and general ontology Contexts Roles (primitives). Aggregation.
* Reify Layers (Data, Schema, Behavior) Meta Model Roles (Metaclass, Class / Instance, Context / Role, Occurrence) as Resources / Kinds Attribute / Values. Resource Values. Template Matching.
* Reify Statements, Kinds, Resources. Templates Matching.
* Reify CSPO.
* Reify Context, Node, Concept, Node. (Statement Types).
* Upper / Matching (Business)
* Units of Measurement (continuos) APIs /  Ontology.
* Discrete (events) APIs / Ontology.
* CDI: Signatures Functional Dataflow Injection / Bus Resolution.
* Aggregation Statements Types:,
* (Kinds / Types, Object / Resource, Predicate / Arc, Object / Resource);
* (Object / Resource, Kinds / Types, Predicate / Arc, Object / Resource);
* Reification. Infer Kinds / Types vía repeated matching of Statements Attributes occurrences. Quads Encoded. Attribute relation: Mapping from and edge of whom an Arc / Predicate is source.
* Encoding: Reification of Statement types to canonical form:
* (Context, Subject, Attribute, Value);
* Aggregate / de Aggregate mappings expansions.
* Operations: Monads, Transforms, Dataflow. Templates / Mappings for processing representations. TODO.
* Services:
* Data Matching Services (Nodes / Resources)
* MVC: Model; Index Service
* DCI: Data; Index Service;
* Schema Matching Services (Kinds / Types)
* MVC: View (flows / prompts); Registry Service;
* DCI: Context; Registry Service;
* Behavior Matching Services (Predicates / Arcs)
* MVC: Controller; Naming Service;
* DCI: Interaction; Naming Service;
* Augmentations:
* Alignment (Data Matching)
* Activation (Schema Matching)
* Aggregation (Behavior Matching)
* Index Service
* Naming Service
* Registry Service
* Encoding / Matching:
* Functional Primitives: (Matching). Graph Shapes Model. Layers. Example: reify / render / match "uncle" relation / "marriage" situation from graph statements.
* Order: Functional Dataflows. Functor / Transform domain / range Activations.
* Relations: Reified Monads / Transforms Mappings (data, schema, behavior) levels. Wrapper Type, Wrapped Type, Wrapped Value, Transform domain / range. To Do.
* Layers Augmentation. Data, Schema, Behavior Layer wise Statement Roles.
* Alignment: Data Matching Augmentation.
* Activation: Schema Matching Augmentation.
* Aggregation: Behavior Matching Augmentation.
* Templates: reify Transforms. Aggregate Template Statements: mappings Resource value domains, predicates, ranges. Matching / Dataflow.
* (Resource, SK, PK, OK);
* Matching / Aggregation: reified statements, kinds. roles, variables, wildcards.
* Matching: Patterns. Template Kinds matches Statements Kinds.
* Matching Aggregation. Apply Expansion: CSPO Transforms Statements results.
* Resources: Statements, Model Layers CSPO Roles. Resource Monad types / values. Uniform transforms API between models roles augmentations browse. (SAILs).
* Models:
* Sets: (Statement, Kind, Attribute, Value);
* Graph: (Context / Type, Resource, Attribute, Value).
* Graph: (Context : Type, Subject : Node, Attribute : Arc, Value : Node);
* Graph: (Resource, Context / Type, Attribute, Value).
* Roles: (Class, Instance, Occurrence, Role);
* Roles: (Metaclass, Role, Context, Instance);
* Discrete Roles: (Relationship : Metaclass, Relation : Role, Kind : Context, Resource : Instance);
* Dimensional Roles: (Dimension : Class, Measure : Instance, Kind : Occurrence, Value : Role);
* MVC / DCI: (Resource, Data, Context : Form, Interaction : Behavior); Layer Templates. Augmentations.
* Augmentations: SAILs / Services
* Template Statements: Matching (Data), Aggregation (Layers), Expansion (Alignment).
* Matching: reified roles class hierarchy. Relations: common superclass. Comparisons / roles.
* Matching: matching reified role classes hierarchy. Relations: common superclass, order. Comparisons / roles.
* Data Activation:
* Augmentation: Sets Activation. Populate Models and their aggregated SPO inputs Statements. Aggregate SPO Kinds: type inference and matching. Render (Type, Resource, Attribute, Value) Statements. Type: Reified Kind (SPO), Attribute / Value of complementary SPO of Resource. Render (Resource, Type, Statement) Statements. Statement: Reified Statement in corresponding set. Data Alignment. Sets / Graph Models.
* Schema: Layers Aggregation.
* Augmentation: Layers Aggregation. Populate Layers and their aggregated schema values context types. Different Models Aggregations yields equivalent Layers Statements shapes arrangements of Aggregations Models Schema roles. Layers Aggregation. Graph / Roles Models.
* Behavior: Layers Alignment.
* Augmentation: Behavior: Alignment. Entail aggregation Layers Statements expanded CSPOs. Entail Data Flow. Functional Data Flow Contexts. Expanded (possible / matchings) Statements traversal. Behavior Alignment. Discrete / Dimensional Models.
* Layers: Models Aggregation. Layer Contexts: upper previous layer Resource value wrapped in Context Resource Monad.
* (Amantes, Pedro, Ama, María);
* (MariaAmadaPor, Amantes, Pedro, Ama);
* (Amar, MariaAmadaPor, Amantes, Pedro);
* (PedroAmaA, Amar, MariaAmadaPor, Amantes);
* (Amor, PedroAmaA, Amar, MariaAmadaPor);
* Inputs: Aggregate SPO into CSPO: Aggregates Contexts Type / Table / Class Kinds. Aggregate PK Cols, Cols : Occurrence, Val : Resources.
* Inputs (Rel / Graph): (Type / Table / Class, PK : Resource, Col : Occurrence, Val : Resource).
* Inputs (Rel / Graph) FKs: Val : Resource equivalent PKs.
* Augmentations / Transforms:
* Data matching. Resource equivalence: identity / comparisons / order transforms.
* Schema matching. Predicates equivalence. Domain / Range types. Order: data flow contexts.
* Behavior matching. Domain / Range values applied functional predicates identity. Order: data flow interactions.
* Transforms: Resource Roles browsing, i.e.: [rsrc.occ.role.rsrcs.ctx.roles](http://rsrc.occ.role.rsrcs.ctx.roles). Resource Roles getters.
* Data layer: matching alignment transforms. [rsrc.type.rsrcs](http://rsrc.type.rsrcs). Resource / Services augmentations.
* Encodings: Ternary. FCA. Graphs. Primes / Bitstrings. Tensors.
* Functional Facades:
* DOM. Data Flow. Functional DOM REST HATEOAS Facades. Browse / Functors.
* Contexts. Transforms.
* Upper / Matching (Business)
* Units of Measurement (continuos) APIs /  Ontology.
* Discrete (events) APIs / Ontology.
* CDI: Signatures Functional Dataflow Injection / Bus Resolution.
* Aggregation Statements Types:,
* (Kinds / Types, Object / Resource, Predicate / Arc, Object / Resource);
* (Object / Resource, Kinds / Types, Predicate / Arc, Object / Resource);
* Encoding / Matching:
* Functional Primitives: (Matching). Graph Shapes Model. Layers. Example: reify / render / match "uncle" relation / "marriage" situation from graph statements.
* Layers Augmentation. Data, Schema, Behavior Layer wise Statement Roles.
* Alignment: Data Matching Augmentation.
* Activation: Schema Matching Augmentation.
* Aggregation: Behavior Matching Augmentation.