* Sets (Quads): Augmentations:
* Aggregation
* Activation
* Alignment
* Functional Sets Relations. Mappings / Transforms.
* Sets (Wrappers):
* Resource : Populate SPO Sets.
* Kind: Resource Context. Aggregate Kinds.
* Statement: Kind Context. Build Statements for each SPO Kind.
* Mapping: Statement Context. Core Model.
* Transform: Mapping Context. Core Model.
* Sets HATEOAS / Data Flow IO Model Statements: (Transform, Mapping, Statement, Kind);
* Core Model: Assert Transform / Query Mapping of Statement Kind Object T. Resulting U : Flows Transforms Statements Kinds matching domains.
* Sets (Wrapped): Dimension, Time, Measure, Employment, etc. Model reified.
* Universe: Resources: (Context : Resource, Subject : Occurrence, Predicate : Attribute, Object: Value);
* Alignment: Aggregate Context Resource Occurrences, Attributes, Values for Resource, Kinds, Statements, Mapings, Transforms Resources from Statement, Mapping, Transforms occurrences / occurring. Functional APIs: Resources Roles Reification. Aggregation: Matching.
* Subjects: (SubjectKind, Subject : Subject, Attribute : Resource P, Value : Resource O);
* Predicates: (PredicateKind, Attribute : Resource S, Predicate : Predicate, Value : Resource O);
* Objects: (ObjectKind, Attribute : Resource P, Value : Resource S, Object : Object);
* SubjectKind (SK): Predicate / Object Intersection. Occurrence:
* (Context : Statement, Subject : SubjectKind, Predicate : Predicate, Object : Object);
* PredicateKind (PK): Subject / Object intersection:
* (Context : Statement, Subject : Subject, Predicate : PredicateKind, Object : Object);
* ObjectKind (OK): Predicate / Subject intersection. Occurring.
* (Context : Statement, Subject : Subject, Predicate : Predicate, Object : ObjectKind);
* Statement / Mapping / Transform: Subject / Predicate / Object intersection:
* SPO Kinds from Kinds Context Statements. Core Statement Inputs / Materialization (Augmentations).
* (Context : Statement, Subject : Subject, Predicate : Predicate, Object : Object);
* (Context : Mapping, Subject : Subject, Predicate : PK, Object : Object);
* (Context : Transform, Subject : SK, Predicate : PK, Object : OK);
* Type Inference: Kinds (Classes):
* Aggregate same Attributes occurrences for sets of Resources sharing same Attributes. Activate Context Transforms Kinds. Activate Kinds Resources Statements.
* Wrapped Types (Kinds) Inference / Matching.
* Core Inputs / Augmentation Flows:
* Add Statements.
* Kinds Activation:
* Aggregate Kinds.
* Aggregate Transforms / Mappings.
* Aggregate SPO Resources from Statements / Transforms / Mappings / Kinds:
* Render / Reify Core Model (Resources, Kinds, Statements, Mappings, Transforms) into Sets:
* Statements, Mappings, Transforms.
* Aligned Statements: Populate Core Model: Templates: Transforms / Mappings.
* Sets Functional Augmentation inference, matching, alignment: data / schema / behavior.
* Resource Hierarchy Categories: Resources, Kinds, Statements, Mappings, Transforms.
* Resources:
* C: Contexts (Transforms / Class), S (Occurring Resource), P (Mapping / Transform), O (Occurrence Resource) Resources.
* Kinds:
* Kinds: SubjectKind, PredicateKind, ObjectKind. Peter valueOf Employee Kind Category Instance.
* Kinds (Wrapped Types / Class). Encoded in Statement Context (Transform / Class). Resolveable Resource Types: Subject SubjectKind (Subject POs) Kind Statement Context.
* Augment Resources with Kinds in Context. Core Model Transforms Mappings Instances / Roles: Kinds and Singleton (Resource) Class.
* Statements:
* Statements : Performed / Matching Contexts Transforms.
* Statements Resources: C: Transform / Class, S (Occurring Resource), P (Mapping / Transform) /  O (Occurrence Resource). S/O. Values: Resources / Reified
* Values. Resources Model / Domains Objects Hierarchy. Activation Augmented Types. Alignment Domain APIs (Measures, Dimensions, etc.).
* Mappings:
* Statement Predicates.
* Context / Statement Transforms / Mappings:
* Dual Occurrence of Transform (Mapping) / Occurring of Mapping (Transform).
* Transforms:
* Context / Transform / Class: (Context, SK, PK, OK). Performable Mappings Templates from learnt input / inferred Statements.
* Reification: Augmented Model.
* Encoding: Representations: Instances / Literals Encoding. URNs. Resolution: sameAs Mappings / Parsing. Occurrence / Occurring domainOf / rangeOf Type Inference.
* Augmentation (Core Model):
* ::augmentedResources
* Activation:
* ::getType (Occurrence)
* ::getSupertype
* ::getSubtypes
* ::getMetatypes (Occurring)
* Alignment:
* ::sameAs
* Aggregation:
* ::parent
* ::previous
* ::next
* ::siblings
* ::children
* Augmentations: Activation (Schema), Alignment (Data), Aggregation (Behavior) Matching (Mapping Function) results: Template Transforms (noop, merge, add); Transforms Flow State: listening for Matching Inputs.
* CSPO Inputs: Wrapper / Wrapped Core Statement Type / Instance Inference (data / schema / behavior):
* Parse into Core Model Templates (Raw Mappings / Transforms to be Augmented / Matched / Merged):
* Inputs Occurrences / Occurring Augmentations: Alignment, Aggregation, Activation of raw Template Inputs. Matchings / Merge (data, schema, behaviors)
* Inputs:
* Augment Resource with Kind in Context. Core Model Transforms Mappings Instances / Roles: Kinds and Singleton (Resource) Class.
* (Class : Transform, Instance : Kind T, Atribute : Mapping, Value : Kind U);
* Inputs Normal Forms: Dimensional, Discrete, etc. Parse Aggregations into Core Model.
* Activation:
* Matching / Templates.
* Infer T / U Resources Kinds Wrapped Types / Instances (Place / NY), Infer Transform Class / Metaclass by SPO Kinds. Parse Instances (Subject / Object Resources) Wrapper / Wrapped Types (Kinds Matching). Transform / Kinds Resolve Mapping Statement (noop, merge, add);
* Matching in Occurrence / Occurring Direction.
* Resource::Mapping::Kind
* Alignment:
* Matching / Templates.
* Assert: (Class / Transform, Resource : Kind T, Attribute / Mapping, Resource : Kind U);
* Query: (Class / Mapping, Resource, Attribute / Transform, Resource) : CSPOs;
* Core Model:
* Occurrence / Mapping Declaration: (Mapping / Class / Metaclass,  Resource / Instance : T, Transform / Occurrence / Context / Statement / Class / Metaclass, Resource / Instance / Role : U);
* Occurrence Object Member of Subject as Transform / Function Role. Instance : T has Member Instance : U.
* Occurring / Transform Application: (Transform / Class / Metaclass, Resource / Instance : Kind T, Mapping / Occurring / Context / Statement / Class / Metaclass, Resource / Instance / Role : Kind U);
* Occurring Subject Member of Object as Mapping / Function Role. Instance : T is Member of Instance : U.
* Aggregation:
* Matching / Templates.
* Aggregation (Layers / Sets) Augment / Encode into Core Model:
* Inputs Normal Forms: Dimensional, Discrete, etc. Parse Aggregations into Core Model.
* (Class : Transform, Instance : Kind T, Atribute : Mapping, Value : Kind U);
* (Time, 1h, mins, 60m);
* (Working, 1h, USD, 40);
* (Working, 160h, USD, ?);
* (Employment, anEmployment, employee, John);
* (Employment, employee, Employee);
* (Employment, anEmployment, employer, ABC Inc);
* (Employment, employer, Employer);
* (John: Transform / Singleton, John, employment, anEmployment);
* (Employee, employment, Employment);
* (ABC Inc: Transform / Singleton, ABC Inc, employment, anEmployment);
* (Employer, employment, Employee);
* (John, employmentAt, ABC Inc.);
* (Employee, employmentAt, Employer);
* (ABC Inc, employsFor, John);
* (Employer, employsFor, Employee);
* Process: Inputs. Core Model (Occurrences / Occurring) Statements From Connectors / IO CSPO:
* Raw CSPO Inputs / Outputs:
* (Class, Instance, Attribute, Value);
* Model Statements I/O.
* Core Model Templates:
* Resources / Predicates Type Inference. Wrapped Types. Transforms / Mappings Types Aggregation. Align. Matching.
* Assert: (Class / Transform, Resource, Attribute / Mapping, Resource);
* Query: (Class / Mapping, Resource, Attribute / Transform, Resource) : CSPOs;
* Layers: Aggregate Inputs into Resources Matrix (Occurences / Occurrings Matrix).
* Perform Augmentations: match / apply Augmentations on each Layer.
* Materialize Augmentations (Occurrences / Occurring Statements). Mappings Transforms Resources Occurrences / Occurring Statements (Models) Results.
* Reactive Data Flow: Process new Models Inputs. Sources: Connectors / Peers, Results Feed Back. Event Sourcing (Models Subscriptions).
* Assert / Query Expansion: State Flows into Contextual Mappings / Transforms.
* Core Models Statements. Mappings / Transforms Declarations / Applications Reifications. CSPO Functional Categories:
* Core Model Statements: Wrapper Category Types:
* Transform : Mapping : Statement : Resource;
* Mapping Category: Query / Browse.
* Query: (Mapping, Resource, Transform, Resource);
* Transform Category: Assert.
* Assert: (Transform, Resource, Mapping, Resource);
* Example: Query / Assert on following Statement upon CSPO Mappings / Transforms Categories.
* Query: (Distance / Mapping, 1km, Meters / Transform, 1000m);
* Assert: (Distance / Transform, 1km, Meters / Mapping, 1000m);
* Occurrence / Mapping Declaration: (Mapping / Class / Metaclass, Resource / Instance, Transform / Occurrence / Context / Statement, Resource / Instance / Role);
* Occurrence Object Member of Subject as Transform / Function Role.
* Occurring / Transform Application: (Transform / Class / Metaclass, Resource / Instance, Mapping / Occurring / Context / Statement, Resource / Instance / Role);
* Occurring Subject Member of Object as Mapping / Function Role.
* Model Aggregation / Expansion (Augmentations match / apply) of Mappings / Transforms Core Statements
* Aggregate into / Expand from Core Model (Layers):
* (Amantes, Pedro, amaA, María);
* (Amantes, María, amadaPor, Pedro);
* Monad Wrapper Role Types Hierarchy:
* Transform : Mapping : Statement : Resource;
* URN : Resource (alignments). Primitives.
* Resource : Root Category. URN : Source / Surrogate Key / Crafted. Naming / Encodings (below).
* Statement : Resource. Wraps Resource as Statement Category.
* Mapping : Statement. Wraps Statement as Functional Transform Declaration Category.
* Transform : Mapping. Wraps Mapping as Functional Transform Application Category.
* Monads Wrapped Roles types / values. Transforms / Mappings. Objects.
* Resource Roles: Reified CSPO Resources types / values, Kinds types / values, Statements, Class, Instance, Occurrence, Occurring, Attribute, Context, Value, Role, Models Roles type / values getters (populated Mappings / Transforms.
* Resources Objects hierarchies / APIs: (Reified Mappings / Transforms): DTOs / Dynamic Functional, DTOs (hashmap) of CSPO roles getters / model domain browsing getters (Functional Transform parameterized Mappings). Flow context: referrers / keys types / values: address::city::street Aggregation / Map. Templates / Mappings: reified declarative Augmentation Data Flows.
* Ontology alignments: Data / Schema / Behavior Augmentations. Model / Schema / Upper / Domains: purposes / gestures (MVC / DCI Mappings / Transforms) layers. Example:
* Transforms: unaEmpresa::unEmpleado::unaPosicion::salary; Salary inferred by context and unaPosicion, unEmpleado, unaEmpresa.
* Class / Instance / Reified (occurrence / ocurring) / Mappings / Transforms:
* Models: Domains / Reified Wrapped Types / Instances. Models, i.e.: Dimensional Domain: Core Model (Transform / Mapping, Resource, Transform / Mapping, Resource) Wrapper Roles, (Dimension, Measure, Unit, Value) Wrapped Instances Types.
* (Distance, 1km, Meters, 1000m); Unit Meters: Occurrence / Occurring Mappings / Transforms Dimensional (upper / aggregated / inferred) Domain Knowledge Assertions Map.
* Statements Augmentation: Dimension::map : Measure, Measure::map : Unit, Unit::map : Values. Wrap C(S(P(O).
* Statement
* Kind
* ContextKind (SubjectKind / ObjectKind flow signature)
* Statement
* SubjectKind
* PredicateKind
* ObjectKind
* Subject
* Predicate
* Object
* Resource
* Context
* Template
* Class
* Instance
* Occurrence
* Value
* Occurring / Context (Statements / Kinds)
* Sets
* Graphs
* Roles (Metaclass, Class, Occurrence, Context, Role)
* Dimensional (Dimension, Measure, Unit, Value)
* Discrete (Relationship, Relation, Kind, Entity)
* MVC / DCI Mappings / Transforms. Example: Forms, Purpose, Gestures, Actors, Roles. Data / Schema / Behavior alignment.
* ESB: Endpoints, Features, Interfaces, Service Process Description / Discovery. Reactive Events Subscriptions. OSGi HATEOAS Endpoints "autowiring".
* BPM: Process, Steps, Flows, etc.
* Augmented Actionable (Process Flows, Items Activation) CMS. Browser: HATEOAS Protocol / APIs / Augmentations. Inferred / Reified / Resolvable Data Flows. Designer: Model Pallete. Declarative core / domains types / instances browsing / discovery "wiring".
* Graph Reified Grammars (upper). Terminal / Non Terminal. Rules / Productions. Mappings / Transform: browse grammar, rules, productions:
* (Rule, Context, lhs, rhs)
* Naming: Kinds / URNs Addressable Encodings. Parsing: URNs Encoded Functional Distributed Resource Resolution. Data Flow Transform / Mappings: Embedded Productions: Augmentations. NLP / NER. Ontology Matching: URN Class Transforms.
* Graph Embeddings: ML Backend Services (ML Predictions Augments Mappings / Transforms). Encodings (Naming).
* Encoding: Deep ML Embeddings. Data: classification, Schema: clustering, Behavior: regression.
* Naming: Auto Encoders. Semantic Hashing. Resources Mappings / Transforms Reified Maps / Tables. Keys / Values Resource Hashing / Resolution Functions: Contextual to Functional Environment State: Mappings Flows / Wrapped State.
* Naming: Augmentations. Contextual Hash Enabled: Functional Mapping Flows Map / Table Encoded / Resolved. Functional Relations: Ontology Matching / Aggregation / Inferences by Hash Encoded Metadata / Transforms Resolutions.
* Clients / Browsers: Peers. Protocol: Reactive Dialogs Prompts. Events. Distributed Data, Schema, Behavior Core Model Statements Encoded I/O: Layers Sync / Augmentation of Knowledge requested from each Peer(s) as Model inputs given resolution of Dialog (Subscriptions) event sourcing state. MVC / DCI Distributed State Transforms / Mappings. Augmented Peer(s) Models: updated View State (flows) / Mappings / Transforms. Rendezvous Peer Role. Local Peer: APIs for local / remote views (MVC / DCI) views (Web, REST) Rendering.
* (...)
* Layers, input layer: Model Roles. Aggregation: Layer Roles shifting until full Layers Roles Statements. Layer: Augments Models.
* Core Statements Roles Resource Interleaved Model (Infer Types, PKs, FKs):
* Occurrence: (Class, Instance, Attribute, Value);
* Occurring: (Class, Instance, Occurrence,  Role);
* Statement Transforms / Relations: order, equivalence, roles, etc.
* Resource of Resource Monad: Occurrence  / Occurrings Quads CSPO Members Aggregation Transforms.
* Resource(t : T) :: contexts :: subjects :: predicates :: objects : Resource(u : U)
* Resource of URNs: Aggregated CSPOs Transforms of Occurrences / Occurrings. Matchings.
* [Resource.of](http://resource.of)(Resource / Class, Instance, Occurring Attribute / Occurring URN, Attribute Value / Occurring Role) :: contexts :: subjects :: predicates :: objects : Resources(CSPO / Resource / occurrence / occurring : URN).
* Resources of Resources: Occurrences / Occurring CSPOs Transforms Matching Wrapped / Wrappings / Transforms Resource Types Shapes Matching Templates. Data flow, apply transforms, order, lists.
* [Resource.of](http://resource.of)(templ : Templ) :: contexts : Resources(c : Context).
* Layer Template Mappings:
* Template : Context : Statement : Resource Layer Roles Monads.
* Layers Quads Aggregation: rotating value role types from previous layer to next layer from V to C):
* 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. Example: Template wraps previous layer Value as Context, Layer Context Value wrapped as Subject value (shifting).
* Core Roles (Wrappers):
* Resource : Monad(x : Resource);
* Statement : Monad(x : Resource) : Resource, Quad DTO.
* Context : Monad(x : Resource) : Statement;
* Template : Monad(x : Resource) : Context.
* Template : Context : Statement : Resource
* Input Layers: Core Statements Roles Resource Shapes: (interleaving graphs models) : Statements CSPO.
* Occurrence Input: (Class, Instance, Attribute, Value);
* Occurring Input: (Class, Instance, Occurrence, Role);
* Output Layer: (Template, Context, Statement, Resource); Input OPSC Resources wrapped in output Layer Roles.
* Sets, Graph, Roles: Services / Augmentations helper Models / Facades.
* Example:
* (Amantes, Pedro, Ama, María);
* (MariaAmadaPor, Amantes, Pedro, Ama);
* (Amar, MariaAmadaPor, Amantes, Pedro);
* (PedroAmaA, Amar, MariaAmadaPor, Amantes);
* (Amor, PedroAmaA, Amar, MariaAmadaPor);
* Augmentations:
* Model Augmentations:
* Alignment: Data Matching. Resources.
* Alignment::match
* Alignment::perform
* Alignment::greaterThan
* Alignment::equals
* Alignment::lessThan
* Activation: Schema Matching. Kinds.
* Activation::match
* Activation::perform
* Activation::superTypeOf
* Activation::sameTypeOf
* Activation::subTypeOf
* Aggregation: Behavior Matching. Contexts Flows.
* Aggregation::match
* Aggregation::perform
* Aggregation::beforeThan
* Aggregation::contains
* Aggregation::containedIn
* Aggregation::afterThan
* Domain Augmentations:
* Transforms Reified in Layers Contexts. Pattern Matching Template Layer resolved:
* Mapping::match
* Mapping::apply
* Mapping::Context
* Mapping::Subject
* Mapping::Predicate
* Mapping::Object
* Encode reified Template Mappings / Transforms. Patterns:
* Model / Domain Augmentations Mappings / Transforms:
* (Wrapper, Wrapped, Mapping, Transform);
* Next Layer step: match / apply Augmentations.
* Layers Template: Layer of CSPO Data Flow Patterns Resolution Resources : Reified (meta) Resources.
* Once Models Layers Matrix are built and populated / aggregated:
* Layers steps Augmentations: perform shifting and wrapping of aggregation values. Layer step Template Mapping: Layer::nextLayer : Layer, for each Layer, match / performs.
* Template, Mapping, Pattern
* (Template, Context, Statement, Resource);
* Resource :: occurrences :: roles :: contexts : Resource