* Model Entities (ModelManager Roles)
  + Embedding : URN + Context Prime ID
  + Kinds : Embeding
  + CSPO Occurrences :  Embedding
  + Statements : Embedding
* FCA Contexts: Axis, Objects, Attributes, Concepts
* Services (FCA / DCI Contexts)
  + Entities (Actors / Roles Occurrences) : Embedding
  + Naming Service : Kinds. (Statement, CSPO) : Kind;
  + Registry Service: CSPO Occurrences. (Statement, Kind) : CSPO;
  + Index Service: Statements. (CSPO, Kind) : Statements;
* Views (Peers) Augmentation Contexts
  + Aggregation Service (Naming : Kinds)
  + Alignment Service (Registry : CSPO Occurrences)
  + Activation Service (Index : Statements / Links Inferences)
* Restful DOM / OGM (DCI, CDI)
* DOM / DCI / MVC (Roles)
* Hierarchies
* Data / Information (Interpretation)
* Knowledge (Information Behavior)
* Roles / Actors (User)
* Definitions
* Implementations
* Relations
* Protocol (View Gestures)
* Contexts (Dispatch). Nested
* Interactions (Roles / Flows)
* Function (Signatures)
* Entity
  + Name
  + Type
  + Value
  + atributes : Set<Entity>
* Type : Entity
  + Name
  + players : Set<Entity>
  + members : Set<Type>
* Role : Type (Wrapper)
  + Type
  + Declarative Behavior / Relations?
* Context : Role (Composite)
  + Entity
  + Type
  + Role
  + Declarative Behavior / Relations?
* Interaction
  + bindings : Set<Context>
* Runtime: OSGI / ServiceMix / Vert.x / Spring Boot / Qi4j / CDI.
* Monad. RxRDF4J. SaILs / Models Services.
* Embeddable / Embeddings (FCA, etc).
* Sets Kinds Model (reifyable).
* Components Architecture: CDI. Endpoints / Routes declarative bindings. Runtime: Vert.x / OSGi, Spring. Discovery / Auto Wiring (signatures).
* Services.
* HATEOAS REST (HAL) Architecture. Components:
* CDI. DDD. MDA.
* DOM. OGM. DCI. MVC.
* Wrapper<? super Embeddable>
* URN / Embeddable: Registry / Factories. Prompts / placeholders. Naming, Index Services.
* Embedding
* Context (Kind / Instance)
* Next Embedding
* Contains Embedding
* Add Embedding
* Meet
* Join
* Concepts / Subsumption
* Inference Association Rules
* Embeddable
* URN
* Embedding (calc from State)
* Statements : Occurrences / Contexts:
* Events: context / occurrence (from Statement events): Update service layers models.
* Service layers:
* Models (for streams). Statement events.
* Augmentation
* Consumes / Produces: onStatement
* Aggregation Service
* Consumes: Resource
* Produces: Kinds
* Activation Service
* Consumes: Kinds
* Produces: Templates
* Alignment Service
* Consumes: Templates
* Produces: Inferred Statements. Entailments. Feedback Aggregation.
* Occurrence statements:
* (Kind, Instance, kindAttribute, kindValue). (Employee, aPerson, Employment, anEmployment);
* Embedding: Kind Context, Instance Objects, discretized attributes. Data. Concepts hierarchy.
* Context Statements:
* (Instance, Kind, kindAttribute, kindValue). (anEmployment, Employment, Employee, aPerson);
* Embedding: Instance Context, Kind Objects, discretized attributes. Schema. Concepts hierarchy.
* Instance : Embeddable;
* Kind : Instance;
* kindAttribute : Instance;
* kindValue : Instance;
* Order, Shapes
* Ontology Matching
* Getters/ Setters
* Available Transforms
* Resource : Embeddable / Addressable (Wrapper?)
* Embeddable : player?
* Map<Kind, List<Statement>> : occurrences?
* Map<Statement, Kind> : occurrences?
* Statement : context
* Kind : role
* Statement : Resource
* Context : Context
* Subject : Subject
* Predicate : Predicate
* Object : Object
* Kind : Statement
* Kind<INST super Resource, ATTR super Resource, VAL super Resource>
* Context : Kind<> (this)
* Subject : INST
* Predicate : ATTR
* Object : VAL
* List<Kind<>> : statements
* Context : Resource (Wrapper)
* Statement : context
* ContextKind : role
* player : Resource
* ContextKind : Context, Kind
* extends Kind<Context, Subject, Predicate>
* Subject : Resource (Wrapper)
* Statement : context
* SubjectKind : role
* player : Resource
* SubjectKind : Subject, Kind
* extends Kind<Subject, Predicate, Object>
* Predicate : Resource (Wrapper)
* Statement : context
* PredicateKind : role
* player : Resource
* PredicateKind : Predicate, Kind
* extends Kind<Predicate, Subject, Object>
* Object : Resource (Wrapper)
* Statement : context
* ObjectKind : role
* player : Resource
* ObjectKind : Object, Kind
* extends Kind<Object, Subject, Predicate>
* Wrapper<? super Embeddable>
* URN / Embeddable: Registry / Factories. Prompts / placeholders. Naming, Index Services.
* Embedding
* Context (Kind / Instance)
* Next Embedding
* Contains Embedding
* Add Embedding
* Meet
* Join
* Concepts / Subsumption
* Inference Association Rules
* Embeddable
* URN
* Embedding (calc from State)
* Statements : Occurrences / Contexts:
* Events: context / occurrence (from Statement events): Update service layers models.
* Service layers:
* Models (for streams). Statement events.
* Augmentation
* Consumes / Produces: onStatement
* Aggregation Service
* Consumes: Resource
* Produces: Kinds
* Activation Service
* Consumes: Kinds
* Produces: Templates
* Alignment Service
* Consumes: Templates
* Produces: Inferred Statements. Entailments. Feedback Aggregation.
* Occurrence statements:
* (Kind, Instance, kindAttribute, kindValue). (Employee, aPerson, Employment, anEmployment);
* Embedding: Kind Context, Instance Objects, discretized attributes. Data. Concepts hierarchy.
* Context Statements:
* (Instance, Kind, kindAttribute, kindValue). (anEmployment, Employment, Employee, aPerson);
* Embedding: Instance Context, Kind Objects, discretized attributes. Schema. Concepts hierarchy.
* Instance : Embeddable;
* Kind : Instance;
* kindAttribute : Instance;
* kindValue : Instance;
* Order, Shapes
* Ontology Matching
* Getters/ Setters
* Available Transforms
* Resource : Embeddable / Addressable (Wrapper?)
* Embeddable : player?
* Map<Kind, List<Statement>> : occurrences?
* Map<Statement, Kind> : occurrences?
* Statement : context
* Kind : role
* Statement : Resource
* Context : Context
* Subject : Subject
* Predicate : Predicate
* Object : Object
* Kind : Statement
* Kind<INST super Resource, ATTR super Resource, VAL super Resource>
* Context : Kind<> (this)
* Subject : INST
* Predicate : ATTR
* Object : VAL
* List<Kind<>> : statements
* Context : Resource (Wrapper)
* Statement : context
* ContextKind : role
* player : Resource
* ContextKind : Context, Kind
* extends Kind<Context, Subject, Predicate>
* Subject : Resource (Wrapper)
* Statement : context
* SubjectKind : role
* player : Resource
* SubjectKind : Subject, Kind
* extends Kind<Subject, Predicate, Object>
* Predicate : Resource (Wrapper)
* Statement : context
* PredicateKind : role
* player : Resource
* PredicateKind : Predicate, Kind
* extends Kind<Predicate, Subject, Object>
* Object : Resource (Wrapper)
* Statement : context
* ObjectKind : role
* player : Resource
* ObjectKind : Object, Kind
* extends Kind<Object, Subject, Predicate>
* Runtime: OSGI / ServiceMix / Vert.x / Spring Boot / Qi4j / CDI.
* Monad. RxRDF4J. SaILs / Models Services.
* Embeddable / Embeddings (FCA, etc).
* Sets Kinds Model (reifiable).
* Components Architecture: CDI. Endpoints / Routes declarative bindings. Runtime: Vert.x / OSGi, Spring. Discovery / Auto Wiring (signatures).
* Services.
* HATEOAS REST (HAL) Architecture. Components:
* CDI. DDD. MDA.
* DOM. OGM. DCI. MVC.
* DOM / DCI / MVC (Roles)
* Hierarchies
* Data / Information (Interpretation)
* Knowledge (Information Behavior)
* Roles / Actors (User)
* Definitions
* Implementations
* Protocol (View Gestures)
* Contexts (Dispatch). Nested
* Interactions (Roles / Flows)
* Function (Signatures)
* Entity
* Name
* Type
* Value
* attributes : Set<Entity>
* Type : Entity
* Name
* players : Set<Entity>
* members : Set<Type>
* Role : Type (Wrapper)
* Type
* Declarative Behavior? / Relations?
* Context : Role (Composite)
* Entity
* Type
* Role
* Declarative Behavior? / Relations?
* Interaction
* bindings : Set<Context>
* Resource embeddings: Resource.of Resource Types (C, S, P, O, Statement, Kind parameterized class). Order. Ontology Matching.
* Resource types implements functional Embeddings (FCA) Interface: (URN, Occurrence: Statement, Kind): EmbeddingID, Aggregated EmbeddingID.
* Resource Types: (URN, Statement, Kind);
* Statement : SK, PK, OK : C, S, P, O : URN : Embedable;
* Kinds; SK, PK, OK.
* Resource Types Transforms:
* ::getKinds / ::getKind (over Context).
* ::getOccurrences : Kinds / Statement of Resource context.
* :: getResource : for Kind in Occurrence Context.
* Statement: getURN/CSPO(Kind) : URN/CSPO;
* Resource(? super Embedable);
* URN : Embeddable.
* Statement : URN
* Context context
* Subject subject
* Predicate predicate
* Object object
* SubjectKind : URN
* Kind ID (SK URN scheme)
* Subject
* Predicate
* Value
* (idem PredicateKind, ObjectKind). Aggregated by Kind ID.
* Subject : URN
* Context : SubjectKind
* URN
* Set(Statement) occurrences
* (idem Predicate, Object). Aggregated by URN.
* Resources Embeddings Registry / Index, Naming (prompts, placeholders). Resolvable URNs.
* URN ID: Contexts. FCA Contexts Prime ID Embedding.
* Augmentations Facade: onStatement feed / stream from DB. Sync RDF4J Models. SaILS / Endpoints layers. Invokes Services.
* Aggregation Service: merge Statements feed with onResource (CSPO, Statement, Kind) parameterized Resource(T) Service feeds. Returns Populated CSPO, Kind, Statements Resources. Repeated invocation augments Resources Embedding.
* Consumes: onResource (C / S / P / O) events stream;
* Creates / Updates CSPOs / Statements Occurrences Kinds Aggregation.
* Produces: Resource(Kind) stream;
* Activation Service: merge Statements feed with onResource Populated Kind Resources (CSPO, Statement, Kind). Returns Statements Kind Resources (Statement Templates: Aggregate Kind Resources by Statements Occurrences). Available Transforms.
* Consumes: Resource(Kind) stream;
* Populates Kinds CSPOs / Occurrences.
* Produces: Kinds CSPO Statements stream;
* Alignment Service: merge Statements feed with Statements Kinds Resources (Kinds Statement Templates). Returns inferred / augmented Statements.
* Consumes: Kinds CSPO Statements stream;
* Populates Statements with matching Statements available Kinds / CSPOs.
* Produces: Resource(Statement) stream.
* Embeddings State: Layers Augments / Updates Resource Embeddings: Aggregation example, Resource(Subject). Multiple invocation of onSubject for same URN returns Augmented Resource(Kind) Kind.
* Feedback Aggregation.
* Layers (RDF4J Sails):
* Semiotic Layer: Verbs, Relationships.
* DOM Layer: Dynamic Object Model. CSPO Materialized Semiotic Layer.
* Sets Layer: DOM Layer CSPO Resource Arrangement.
* Augmentations / FCA Layer: Sets Layer FCA / Embeddings.
* Functional Layer: API.
* Semiotic Layer:
* Verbs: action (rel end: amante) / passion (rel end: amado) / state (rel: amor: ama / amado). Relation parts attributes.
* Semiotics: CSPO Schema. DOM Statements Source.
* Occurrences: Objects.
* Concepts: Non Terminals.
* Signs: Terminals.
* Semantics / Pragmatics:
* (Context, Occurrence, Sign, Concept);
* (Context, Occurrence, Concept, Sign);
* Grammar:
* (Context, Concept, Occurrence, Sign);
* (Context, Concept, Sign, Occurrence);
* Syntax:
* (Context, Sign, Concept, Occurrence);
* (Context, Sign, Occurrence, Concept);
* DOM Layer:
* (Class, Instance, Attribute, Value);
* Sets Layer:
* Sets Encoding / Arrangements:
* Subject, Predicate, Object Sets inside Context Set (CSPO Statements Resources Populated)
* SubjectKind Set: Statements Predicates / Objects intersection.
* PredicateKind Set: Statements Subjects / Objects intersection.
* ObjectKind Set: Statements Predicates / Subjects intersection.
* Kinds Population (Subject Example): Aggregate Subjects Occurrences with same Predicates (type), same Objects (instance).
* Kind Reification (Subject Example): S: (SubjectKind, Subject, Predicate, Object);
* Resource Monad: For each input SPO URNs, prime sequence identifying each new Resource. Resource registry.
* Resource Embeddings: set of Resources Occurrences (Statements) Contexts / Attributes products.
* Statements: SPO / FCA (Object, Context, Attribute) Resource triples.
* Kind Monad: For each Kind set of matching Attributes / Values, set of matching Resources. Identifier (reification) Resource: Embedding URN.
* CSPO quads Statements Context: Kind (inheritance).
* Resource<URN>
* Resource<Statement>
* Resource<Kind>
* SubjectKind: Kind;
* PredicateKind: Kind;
* ObjectKind: Kind;
* Resource<URN>::map(ctx::getSubjectOccurrences) : Set<Resource<Statement>>;
* Resource<URN>::map(ctx::getPredicateOccurrences) : Set<Resource<Statement>>;
* Resource<URN>::map(ctx::getObjectOccurrences) : Set<Resource<Statement>>;
* Resource<Statement>::map(ctx::getSubjectKind) : Resource<Kind>;
* Resource<Statement>::map(ctx::getPredicateKind) : Resource<Kind>;
* Resource<Statement>::map(ctx::getObjectKind) : Resource<Kind>;
* Resource<Kind>::map(ctx::getKindContexts) : List<Resource<URN>>;
* Resource<Kind>::map(ctx::getKindAttributes) : List<Resource<URN>>;
* Resource<Kind>::map(ctx::getKindValues) : List<Resource<URN>>;
* Augmentations / FCA Layer:
* FCA:
* FCA Attributes: Primes / one-hot Bitstring Encoding.
* FCA Context: Concepts / Objects Attributes.
* Embeddings: CSPOs FCA Contexts Objects URNs Attributes Primes Product / Bitstring OR.
* Vector Space Model: CSPO Dimensions. CSPO Points: Objects URN Embeddings.
* Similarity / Distance: Common FCA Embedding Attributes Factors. Common Super Concept / Object. VSM Vector Similarity.
* Transforms / Translation: Object, Object. Merge Attributes, extract Similarity on merged Subject and merged Objects (Transforms).
* State (flow): Attributes index (hasAddress), values (address: xyz) masks browsing (Concepts / Objects).
* Ontology Matching: Concept Lattice shape.
* Augmentations:
* Generate Embeddings from DOM SPO URNs FCA Contexts (Context(Attributes, Objects)).
* FCA Contexts: Map Attributes URNs with primes sequence / one-hot bitstring and Objects URNs with theirs Attributes product / bitstring OR Embedding. Embed Context (SPO / Kinds) into Contexts Attributes (product).
* SPO URNs Resources Wrapper (Contexts Attributes / Objects Embeddings Population / Encoding). Subject URN Embedding Encoding Example: Aggregate all Subject Embedding Contexts Occurrences Concept Attributes Primes Product.
* Subject Embeddings: Contexts: Predicates; Attributes: Objects; Objects: Subjects;
* Predicate Embeddings: Contexts: Objects; Attributes: Subjects; Objects: Predicates;
* Object Embeddings: Contexts: Subjects; Attributes: Predicates; Objects: Objects;
* SubjectKind Embeddings: Contexts: Subjects; Attributes: Predicate Contexts (occurrences for Subject); Objects: Object Contexts (occurrences for Subject);
* PredicateKind Embeddings: Contexts: Predicates; Attributes: Subject Contexts (occurrences for Predicate); Objects: Object Contexts (occurrences for Predicate);
* ObjectKind Embeddings: Contexts: Objects; Attributes: Predicate Contexts (occurrences for Object); Objects: Subject Contexts (occurrences for Object);
* Schema Aggregation: Type (Kind) in Context / Role Inference. Align Attributes with existing Kinds. CSPO Embeddings Clustering. Kinds Naming / Labels: Alignment Embedding Encoded Placeholders.
* Data Alignment: Align / Reify Instances with Kinds. Complete (align) Type (Kind) Instance Attributes Values in Context / Role Inference. Embeddings Features zero-shot Classification (Aggregation Kinds Embedding Encoded Labels). Missing Values (links) Prediction: Placeholder Embeddings with resolvable context metadata.
* Behavior Activation: Performed / Available Resource State (DIDs) Transforms. Perform Prompt: Aggregate / Align. Response: Embeddings Context Facts / Next Available Prompts (Alignment). Embeddings Features Regression (Prompts Suggestions).
* Embeddings Features Average (User Embedding, Product Embedding) Predictions.
* Functional API:
* Runtime: RDF4J.
* REST Browsing State Based Dialog Wizard. DOM / DCI / CDI / Augmentations (FCA). Structured Prompts / Responses (Statements Flow, Relationships / Roles).
* URNs: Semantic Identifiers. Encoding. DIDs (Distributed IDs). Angular Encoding.
* Conversational State Transfer (COST): Distributed (P2P).
* Node.js: JSON-LD. Functional Resources (Monads, run-at: request client / server peer. Context State available Functors / Transforms).
* Parsing: Transforms.
* Resource URN Monads: Activation, Alignment, Aggregation. Class / Instance State (Embeddings) Statements.
* URNs Class Statements (Instance, Class, Attribute, Value). Instance in this Class (role, Object Statement) Attributes / Values.
* (Person, Employee, Employer, Enterprise);
* (aPerson, Addressable, Address, anAddress);
* URNs Object Statements (Class, Instance, Attribute, Value). Class in this Instance (occurrence, Class Statement) Attributes / Values.
* (Employee, aPerson, Employer, anEnterprise);
* (Addressable, aPerson, Address, anAddress);
* Layers:
* Schema Activation
* Each URN has its own Class / Instance Attribute / Value Mapping Statements. Parse / Materialize CSPO (Schema / Template). I/O / Activation
* Sets Aggregation
* Each URN has its own Kinds Class / Occurrences Instances Sets Statements.
* FCA Contexts Alignment
* Alignment: Each URN has its own Class / Instance FCA Contexts. Attributes / Values / Concepts: Classes / Instances
* Attributes / Values: Classes / Instances. Singleton Class, class with only one Instance Member. Class : Instance.
* Object Statements Attributes / Values: Role / Player
* Class Statements Attributes / Values: Context / Occurrence
* Data flow:
* Activation, Aggregation, Alignment, Aggregation, Activation.
* Monad Functional: Resource (Resource): Resource (Context). Statements state (Embeddings):
* Object (Attribute): Concept (Kind) / Object (Context)
* Pedro (Empleador): José (Empleo)
* Pedro (José): Empleador (Empleo)
* Empleador (Pedro): José (Empleo)
* José (Empleado): Pedro (Empleo)
* José (Pedro): Empleado (Empleo)
* Empleado (José): Pedro (Empleo)
* Empleado (Posición): Líder (Empleo)
* Líder (Empleado): Posición (Empleo)
* Verticles (Actors), Services, Commands / Events. Messages (encoding / dispatch). Endpoints: URLTemplates + config (URN Streams selectors / handlers / flatMap Activation: behavior schema interfaces).
* RDF Store SaILs, Sets, FCA Contexts, pluggable Services interface discovery.
* Statements streams: Topics. sources / sinks. Subjects. Store / Services. Augmentations. Group By. Map / FlatMap. Filter. Map / Reduce. For each. Order (behavior, state flow).
* Has URN: SPO, Kind (C), Statement / Triple. URN: IRI + Prime ID + Embedding.
* Resource: Monad(Has URN).
* Has URN: Named. Naming: Prompts / Placeholders (Activation) for Kind names.
* Resource (SPOs, Kinds): Function. URN: Embeddings. Resource: Subject (Contexts sub, Function pub). Lift. Observable en vez de callback. Command as Observable.
* Source / Materialized Statements. Feedback: Contexts / Embeddings Dataflow.
* Functional Prompts / Responses: Object / Class Statements.
* Kinds: C is SK, PK, OK Tag, SPOs Kind URN.
* Ontology matching / alignment.
* Augmentations:
* Aggregate Kinds: Schema. Ps: Kind fields. Os: Kind state /metaclass. Kinds: Set Comprehensions. Sets Model / Service / SaIL.
* Align Statements: Data. Values (links prediction). Embeddings. FCA Model / Service / SaIL.
* Activate Resources: Embeddings, Behavior. Functional state flows. DCI / DOM Model / Service / SaIL.
* Streams: Sevices / API Gateway / Facade.
* Stream is Resource Occurrences.
* Statements Contexts. Subjects.
* Kinds (C), SPOs Streams.
* SPO Occurrences: Statements, Kind in Statements.
* Kind Occurrences: SPOs, Statement Context.
* Registry: URN / Prime ID / Current Embedding. Occurrences. Events Sourcing.
* Services. Index. Extract Resouces interfaces / features (behavior / schema). Data.
* Contexts: Calculate Embeddings. Registry, Services, Index: Resolve Embeddings. Functional dataflow update Contexts / Embeddings.
* Sets Contexts. Kinds: Sets Comprehensions.
* FCA Contexts. Objects, Attributes, Concepts (Kinds).
* Observable / Observer (subscribe).
* Subject: Publisher / Subscriber.
* Flowable.fromIterable: map, collect, subscribe, etc.
* Events.
* Topics.
* Queues.
* Resource URN Monads: Activation, Alignment, Aggregation. Class / Instance State (Embeddings) Statements.
* URNs Class Statements (Instance, Class, Attribute, Value). Instance in this Class (role, Object Statement) Attributes / Values.
* (Person, Employee, Employer, Enterprise); (aPerson, Addressable, Address, anAddress);
* (aPerson, Addressable, Address, anAddress);
* URNs Object Statements (Class, Instance, Attribute, Value). Class in this Instance (occurrence, Class Statement) Attributes / Values. (Employee, aPerson, Employer, anEnterprise); (Addressable, aPerson, Address, anAddress);
* (Employee, aPerson, Employer, anEnterprise);
* (Addressable, aPerson, Address, anAddress);
* Layers:
* Schema Activation
* Each URN has its own Class / Instance Attribute / Value Mapping Statements. Parse / Materialize CSPO (Schema / Template). I/O / Activation
* Sets Aggregation
* Each URN has its own Kinds Class / Occurrences Instances Sets Statements.
* FCA Contexts Alignment
* Alignment: Each URN has its own Class / Instance FCA Contexts. Attributes / Values / Concepts: Classes / Instances
* Attributes / Values: Classes / Instances. Singleton Class, class with only one Instance Member. Class : Instance.
* Object Statements Attributes / Values: Role / Player
* Class Statements Attributes / Values: Context / Occurrence
* Data flow:
* Activation, Aggregation, Alignment, Aggregation, Activation.
* Monad Functional: Resource (Resource): Resource (Context). Statements state (Embeddings):
* Object (Attribute): Concept (Kind) / Object (Context)
* Observable.lift(ResourceMonad).
* Pedro (Empleador): José (Empleo)
* Pedro (José): Empleador (Empleo)
* Empleador (Pedro): José (Empleo)
* José (Empleado): Pedro (Empleo)
* José (Pedro): Empleado (Empleo)
* Empleado (José): Pedro (Empleo)
* Empleado (Posición): Líder (Empleo)
* Líder (Empleado): Posición (Empleo)
* Classification
* Regression
* Clustering / Feature Extraction
* Image Classification
* Object Detection
* OCR
* Barcodes
* Face Detection
* SemanticWebAlignmentTheory : Ontology Matching.
* JavaScript Monads.
* User Modeling
* DOM / DCI / MVC (Roles)
* Hierarchies
* Data / Information (Interpretation)
* Knowledge (Information Behavior)
* Roles / Actors (User)
* Definitions
* Implementations
* Relations
* Protocol (View Gestures)
* Contexts (Dispatch). Nested
* Interactions (Roles / Flows)
* Function (Signatures)
* Entity
* Name
* Type
* Value
* atributes : Set<Entity>
* Type : Entity
* Name
* players : Set<Entity>
* members : Set<Type>
* Role : Type (Wrapper)
* Type
* Declarative Behavior / Relations?
* Context : Role (Composite)
* Entity
* Type
* Role
* Declarative Behavior / Relations?
* Interaction
* bindings : Set<Context>
* Runtime: OSGI / ServiceMix / Vert.x / Spring Boot / Qi4j / CDI.
* Monad. RxRDF4J. SaILs / Models Services.
* Embeddable / Embeddings (FCA, etc).
* Sets Kinds Model (reifyable).
* Components Architecture: CDI. Endpoints / Routes declarative bindings. Runtime: Vert.x / OSGi, Spring. Discovery / Auto Wiring (signatures).
* Services.
* HATEOAS REST (HAL) Architecture. Components:
* CDI. DDD. MDA.
* DOM. OGM. DCI. MVC.
* Wrapper<? super Embeddable>
* URN / Embeddable: Registry / Factories. Prompts / placeholders. Naming, Index Services.
* Embedding
* Context (Kind / Instance)
* Next Embedding
* Contains Embedding
* Add Embedding
* Meet
* Join
* Concepts / Subsumption
* Inference Association Rules
* Embeddable
* URN
* Embedding (calc from State)
* Statements : Occurrences / Contexts:
* Events: context / occurrence (from Statement events): Update service layers models.
* Service layers:
* Models (for streams). Statement events.
* Augmentation
* Consumes / Produces: onStatement
* Aggregation Service
* Consumes: Resource
* Produces: Kinds
* Activation Service
* Consumes: Kinds
* Produces: Templates
* Alignment Service
* Consumes: Templates
* Produces: Inferred Statements. Entailments. Feedback Aggregation.
* Occurrence statements:
* (Kind, Instance, kindAttribute, kindValue). (Employee, aPerson, Employment, anEmployment);
* Embedding: Kind Context, Instance Objects, discretized attributes. Data. Concepts hierarchy.
* Context Statements:
* (Instance, Kind, kindAttribute, kindValue). (anEmployment, Employment, Employee, aPerson);
* Embedding: Instance Context, Kind Objects, discretized attributes. Schema. Concepts hierarchy.
* Instance : Embeddable;
* Kind : Instance;
* kindAttribute : Instance;
* kindValue : Instance;
* Order, Shapes
* Ontology Matching
* Getters/ Setters
* Available Transforms
* Resource : Embeddable / Addressable (Wrapper?)
* Embeddable : player?
* Map<Kind, List<Statement>> : occurrences?
* Map<Statement, Kind> : occurrences?
* Statement : context
* Kind : role
* Statement : Resource
* Context : Context
* Subject : Subject
* Predicate : Predicate
* Object : Object
* Kind : Statement
* Kind<INST super Resource, ATTR super Resource, VAL super Resource>
* Context : Kind<> (this)
* Subject : INST
* Predicate : ATTR
* Object : VAL
* List<Kind<>> : statements
* Context : Resource (Wrapper)
* Statement : context
* ContextKind : role
* player : Resource
* ContextKind : Context, Kind
* extends Kind<Context, Subject, Predicate>
* Subject : Resource (Wrapper)
* Statement : context
* SubjectKind : role
* player : Resource
* SubjectKind : Subject, Kind
* extends Kind<Subject, Predicate, Object>
* Predicate : Resource (Wrapper)
* Statement : context
* PredicateKind : role
* player : Resource
* PredicateKind : Predicate, Kind
* extends Kind<Predicate, Subject, Object>
* Object : Resource (Wrapper)
* Statement : context
* ObjectKind : role
* player : Resource
* ObjectKind : Object, Kind
* extends Kind<Object, Subject, Predicate