* Functional Resources / Sets : URI Monad
* Resource URI Monad wraps case classes (URIs: Statement Occurrences Roles):
* Resource::getKind
* Resource::getObject
* Resource::getAttribute
* Resource::getValue
* Context (Kind, Object, Attribute, Value). O,A,V : SPO : Resources / Kinds (schema / metaclass / order / hierarchies).
* Subject (C, S, P, O)
* Predicate (C, P, S, O)
* Object (C, O, P, S)
* Kinds: Type Inference. URIs: Aggregated Resource Object Role Attributes / Values, reified Context URIs IDs (Statement cases)
* SPOs Resource Statements, i.e.: Subject Resource Statement Subject Role Statement: Aggregated Statement Occurrence in Kinds Hierarchy Context Order: single statement / married statement
* S1 : (Single, John : S2, :maritalStatus, Single);
* (Married, John : S1, :maritalStatus, Married);
* Inferred from Aggregated SPOs Statements SPOs Roles Kinds Contexts Occurrences Order (i.e. SPO Roles Occurrences Kinds SPO Statements: SPO Statements for each Statement SPO Occurrence Context):
* (Married, John : S1, :wife, Mary);
* Statements SPO Occurrences Roles linked Metadata Statements. I.e.: SPO Statements of Predicate of Context Statement Predicate Occurrence:
* :maritalStatus :marriedWith Married
* :marriedWith :maritalStatus Married
* Married :maritalStatus :marriedWith
* Married :marriedWith :maritalStatus
* Kind Roles. I.e.: PredicateKind as Subject / Object. Schema.
* Reified Kinds. I.e.: Subject (SuperKind, SubjectKind, Attribute, Value). Hierarchies / Composition / Context Roles / Order: attributes / recursion aggregation.
* Kind Roles (Schema). FCA Contexts Encoding. Lattice / Context: Order / Matching Alignment
* ContextKind
* SubjectKind: Subject Role, Predicate Attribute, Object Value. Schema (implements) : Predicate, Object
* PredicateKind: Predicate Role, Subject Attribute, Object Value. Schema (implements) : Subject, Object
* ObjectKind: Object Role, Predicate Attribute, Subject Value. Schema (implements) : Predicate, Subject
* Statement (Data, Mapping : Interface, Transform : Expression). URIs: Contexts (Kinds / Types). Schema: reified Kinds
* Input Statements case matching yields URI Resource(s) Monads wrapping matching SPO case classes instances in Statement Occurrence Context. Role in Statement: case class given (Context). URI: Statement case class role instance URI. Kinds case class matching aggregates reified Kind Context URIs Object's Attributes / Values in corresponding Kind case classes.
* Crossword Cube Encoding (X:S, Y:P, Z:O).
* Contexts: X(Y, Z); Y(X, Z); Z(Y, X); Kinds / Patterns Matching. Augmentations.
* Statements: State (Dimensional, 4th dimensionalism). Relationships (n-ary).
* Inputs: d2rq, any23, DBPedia (NLP NER for type contexts.
* Deploy: input resources, augmentation, matching, dataflow: functional / streams OGM (OpenRDF Sesame Elmo), qi4j DCI / DOM SaILs.
* Ordenar: timestamp stream.
* Ordenar: ordership relations (A is 2 (units / other relations) before / after B). Causal / effect. State flows. Inclusion (contexts).
* Interfaces (sets):
* Resource : URN, (Resource, Resource, Resource, Resource);
* Getters::CSPO.
* Context : (Context, Resource, Resource, Resource) : Resource;
* Subject : (Resource, Subject, Resource, Resource) : Resource;
* Aggregation: URN (URN, yx  occurrence context, Kind, Resource value);
* Kind : (Kind, Subject, Attribute, Value); Parameterized roles attributes / values.
* SubjectKind : Subject<attr Predicate, val Object> (parent : Subject, SubjectKind, Attribute, Value) : Kind, Subject (reified Context);
* Getters::Type, Resource, Attribute, Value.
* Statement : (Context, Subject, Predicate, Value). Reified Kinds Contexts.
* Deployment: FCA URN Encoding (Distributed Homogeneous URI Monad Contexts) URN Matching: Dynamic Distributed Event Driven Log, encoded contexts relations, incremental metadata. Kinds, Resources aggregated in URN IDs.
* Deployment: URI URN Monad. Wraps URI Monad Occurrences as aggregated URI Resource(s) URN Monad (URI Monad case classes: Occurrences Roles)
* Deployment: URN Monad Endpoints as Vert.x Verticles. Dispatcher Endpoint builds Verticle Network (observer, mappings, transforms) of URI Resource case matching (Verticle types) URN Monad (Verticle contents)
* Augmentation
* Aggregation
* Alignment
* Activation
* Protocol / API: DCI. DOM OGM
* Application: DDD / Profiles
* Ontology Matching / Purpose Alignment
* Profile App:
* Social exchange network. Purpose driven assets management and Collaboration Tool.
* Metaclass, Class, Instance, Context, Occurrence, Role Resource Metada.
* Asset, Need, Good. Purpose.
* Role (discrete) relationships. Translation (actors context) entailments.
* CSPO, Kinds. Statement, Context, Mapping quads. Sets entailments.
* Dimensional (continuous) Relationships. Translation (contexts) entailments.
* Measure / Value (Dimension / Unit) Pairs. FCA Contexts (scaling) inferences entailments.
* Protocol: Semantic Identifiers. URNs.
* Protocol: Context Driven Interaction REST P2P (SIDs URNs: Resources  DCI Dialogs). Runat peer resolution addressable / browseable Messages interactions: request / response Message streams DCI dialogs. Embedded session semantics: event sourcing / history terms / roles resolution / navigation).
* Messages: SIDs URNs Case Classes Statements. Statement Data Pattern Matching. State Flows: Reactive Events Messages.
* Protocol: SIDs URNs Resources. Endpoints: Case Classes Aggregated Message Signatures, Aligned Statements Data Pattern Matching Message Events Resource Statement Occurrences. Resource Monad.
* Core Model Upper Resources (DCI Context / Facets: Metaclass, Class, etc. as Resource, root navigation Context Resource). Aggregation (schema cases) / Alignment (resource statements occurrences): Activation.
* Core Model Functional Transforms: Functional Activation Statements:  Aggregation Schema Case Classes Statements / Alignment Message Events Resource Statement Occurrence.
* Protocol: GET URN Case Classes (Aggregation) / Statements Data (Alignment) Message Events Resource Statement Occurrences.
* Protocol: GET Browse Resource Aggregated / Aligned Message Events Resource Statement Occurrences. Build Context State Flows (Monad Functional Activation).
* Protocol: POST URN Navigation Context State Built Resource Activation Data Statements.
* Protocol: POST Subsequent entailed Context Browsing / Events Functional Transforms Activations.
* Monad: Resources (Metaclass, Class, etc.). Context.
* Transform: Statements (schema and occurrences).
* Sample Workflow:
* aResource.flatMap(anStatement) : aResourceOccurrence;
* aResourceOccurrence.flatMap(Activation::KindsCase) : aKindResource;
* Activation::[Role]OccurrencesCase \*: Occurrence[Role][];
* DCI / MVC DDD Application Layer: OGM (Sesame Elmo / Alibaba. Qi4j). Core / Domains ontologies.
* Resource: Types hierarchies / instances / occurrences (URN, Statement, CSPORole, Kind).
* Relationships (discrete / continuous). Order. Translation / Equivalences entailments.
* Metaclass, Class, Instance, Context, Occurrence, Role Resource Metadata Maps Monad with contextual CSPOs Statements (schema and occurrences) for Resources in Roles.
* Resources: John, Peter, Mary, loves, friendOf, loverHasFriend.
* Transform / Mapping: John :loves Mary;
* Transform / Mapping: Peter :friendOf John;
* Transform / Mapping (Expanded Knowledge): Mary :loverHasFriend Peter;
* Browse Resources: Functional Activation: Transforms / Mappings Contexts Knowledge Expansion. Idem for Kinds and Schema Statements Aggregations / Alignments.
* Expanded Knowledge: Concrete and Navigation Context entailed / materialized Schema and Occurrences Statements.
* Functional Relation Predicates:
* FCA / TMRM:
* Lover(John, Mary);
* Loved(Mary, John);
* Love(Lover, Loved);
* State(Action, Passion);
* Action / Passion: Resource / State Roles.
* State: Context (Action / Passion Pairs. Verbs).
* Resource Roles: Monads.
* State Roles: Mappings / Functional Transforms (Stateful Contexts Browsing).
* Entailments: Switch Cases encoded as Resource Monads Triples. Mappings / Transforms reification:
* Switch (Developer) case: devel.cat = junior, salary: 1000, case devel.cat = semisr: salary: 2000, etc;
* Switch (Designer) case: design.cat = junior, salary: 1000, case design.cat = semisr: salary: 2000, etc;
* FCA: Build Contexts from switch (objects), cases (attributes / values: scaling).
* Case Classes: Verticles Event Bus Pattern Matching (Dispatcher / Signatures).
* Switch Case Patterns: CSPOs, Kinds, Statements Layers Networks. Aggregation, Alignment, Activation (Map Reduce).
* Verticles: URNs (CSPOs, reified Kinds, reified Statements) Monads. Encoding: extract Resources / Roles in URNs Functional Contexts Roles.
* Reactive / Event Driven: Verticles DIDs (Distributed IDs) distributed Resource / Applicable events logs. Rx Facade. Resource URNs Verticle Resolution, Transforms ordered Mappings Statements.
* CSPO Cases: one for each Statement CSPO destructuring case (for each CSPOs Data Aggregation). SCase, PCase, OCase yielding corresponding Verticle Monads.
* Kinds Cases: One for each Kind Type Data / Mappings Agreggation. SKCase, PKCase, OKCase yielding corresponding Verticle Monads.
* Statement Cases: CSPOs / Kinds Aggregation Function (Map Reduce). Entailments.
* Mapping Cases: Statements Aggregation Function (Map Reduce). Entailments.
* Transforms Cases: Mappings Aggregation Function (Map Reduce). Entailments.
* Pattern Matching: Case Matching CSPOs, Kinds Types / Instances Matching. Yields corresponding Monad Type / Instance Wrapper Verticle.
* Pattern Matching: Case Matching Statement, Mapping, Transform Types / Instances Matching. Yields corresponding Monad Type / Instance Wrapper Verticle.
* Monads Verticle Endpoints (topic) consumes Monads, produces available Functions in Monad Functional Context,
* Monad Consume Cases: Instantiate Function Verticles.
* Function Verticle Endpoints (topic) consumes Functions in Monad Functional Context, produces Monads available for Function Application.
* Function Consume Cases: Instantiate Monad Verticles.
* Resource Monad Types:
* SPOs. Stream: Statements (Context Occurrences).
* Resource Monad : CSPOs (URNs Resource Roles in Contexts)
* Kinds. Stream: Mappings (Resource Occurrences).
* Resource Monad : Kinds (SK, S, Attribute, Value) Kind Relative Resource Roles in Contexts.
* Statement: Statement (D) Resource CSPOs.
* Resource Monad : Statements (Resource Quads). CSPO Resource Context Roles.
* Statement: Mapping (C) Resource Kinds.
* Resource Monad : Mappings (Resource Quads). Kinds Resource  Context Roles.
* Statement: Transform (I) Resource Quad Contexts.
* Resource Monad : Transform (Resource Mappings). Resource  Context Roles.
* Quad Contexts. Stream: Transforms (Interactions).
* Functions:
* Applicable
* Mapping Match / Apply Transform.
* Transforms Order (Functional Context Roles):
* (C, CPrevResource, CMappingResource, CNextResource);
* Inferred / Stated. Entailment (Pattern Matching)
* Java pattern matching statements case classes. Resource Monad hierarchy wrapping Case classes.
* Inputs / Sync Adapters:
* (Class, Instance, Member, Value) Events / Messages.
* Data Modelling Resources / Patterns (to / from Adapter Events):
* (Metaclass, Class, Instance, Context, Role, Occurrence);
* Models Patterns:
* (n-ary) Relations / Relationships. Foreign Keys. Dimensional / Measures, Master / Detail, Item / ItemDescription, Contexts / Roles, Interactions / Actors, Model / Facets (Actors / Roles Bindings. Property Graph, Others.
* Switch actions: Populate Models (RDFS, OWL, Sets, FCA Contexts, Functional MVC / DCI DOM / Others: SaILs). URNResource / Contexts / Roles.
* (Dimension, Unit, Measure, Value);
* Equivalence: Same Distance Measures.
* Entailment: Dimension, Unit, Measure Values entails other Measures / Values (Time, Speed, Distance). Contexts / Mappings.
* Composition of Case classes instances Augmentation via pattern matching (Aggregation: Kinds, Alignment: Contexts, Activation: Interactions). Map Reduce.
* Order: Kinds Hierarchies. Contexts Bindings Availability (Interactions State).
* Layers Case Classes Patterns / Resource Monads hierarchy:
* URN : (Class, ID);
* Class : Dimension, Unit (Attribute);
* ID : Measure, Value (Value);
* CSPOs : URN;
* URNResource : FunctionalResource (wrapper Monad).
* Transforms over URNResource wrapped URNs:
* C, S, P, O Case Classes:
* Context : (URN, \_, \_, \_) : URN;
* Subject : (\_, URN, \_, \_) : URN;
* Predicate : (\_, \_, URN, \_) : URN;
* Object : (\_, \_, \_, URN) : URN;
* CSPOResource : FunctionalResource (wrapper Monad).
* Transforms over CSPOResource wrapped Resources:
* I/O Resource Events:
* Resource : (URN, Statement, Attribute, Value) : Statement;
* SubjectResource : (URN, Statement, Predicate, Object) : Resource;
* PredicateResource : (URN, Statement, Subject, Object) : Resource;
* ObjectResource : (URN, Statement, Predicate, Subject) : Resource;
* ResourceMonad : FunctionalResource (wrapper Monad).
* Functional Transforms over ResourceMonad wrapped Resources:
* Input Data Events: Tabular, Dimensional.
* Order. Equivalences / Entailments.
* Augmentations:
* Aggregation : Classification. Kind / Context.
* Attribute / Value MapReduce Aggregation. Order (subset / superset relation)
* Kinds Aggregation Statement: TODO
* Contexts Aggregation Statement: TODO
* Alignment : Clustering: Occurrence / Mapping. Metaclass, Class, Instance, Context, Occurrence, Role Modelling Patterns (Master / Detail, DCI, etc.).
* Occurrence Alignment Statement: TODO
* Mapping Alignment Statement: TODO
* Activation : Regression: Relationship. Distance(Time, Speed). Salary(Position, Expertise).
* Relationship Activation Statement: TODO
* Materialize Augmentation Statements (Resources). Augmented Data Events (Feedback)
* Functional Traversal APIs.
* TODO:
* (URN, ParentAxis, Previous, Next);
* Statement Case Classes:
* Statement : (URN, URN, URN, URN) : URN;
* CSPOStatement : (Context, Subject, Predicate, Object) : Statement;
* StatementResource : FunctionalResource (wrapper Monad).
* Transforms over StatementResource wrapped Statements:
* Subject(Predicate) : Object(s);
* Subject(Object) : Predicate(s);
* Predicate(Subject) : Object(s);
* Predicate(Object) : Subject(s);
* Object(Subject) : Predicate(s);
* Object(Predicate) : Subject(s);
* getContexts
* getSubjects
* getPredicates
* getObjects
* SPO Kinds Resource Case Classes:
* Kind : (URN, Resource, Attribute, Value) : Statement;
* SubjectKind : (URN, SubjectReifiedKind, Predicate, Object) : Statement;
* PredicateKind : (URN, PredicateReifiedKind, Subject, Object) : Statement;
* ObjectKind : (URN, ObjectReifiedKind, Predicate, Subject) : Statement;
* KindResource : Functional Resource (wrapper Monad).
* Transforms over (higher kinds reified) KindResource wrapped Statements:
* SubjectKind(PredicateKind) : ObjectKind(s);
* SubjectKind(ObjectKind) : PredicateKind(s);
* PredicateKind(SubjectKind) : ObjectKind(s);
* PredicateKind(ObjectKind) : SubjectKind(s);
* ObjectKind(SubjectKind) : PredicateKind(s);
* ObjectKind(PredicateKind) : SubjectKind(s);
* getSubjectKinds
* getPredicateKinds
* getObjectKinds
* Contexts Case Classes:
* Context : (URN, Kind, Kind, Kind) : Statement;
* SubjectContext : (URN, SubjectKind, PredicateKind, ObjectKind) : Context;
* PredicateContext : (URN, PredicateKind, SubjectKind, ObjectKind) : Context;
* ObjectContext : (URN, ObjectKind, PredicateKind, ObjectKind) : Context;
* ContextResource : FunctionalResource (wrapper Monad).
* Transforms over ContextResource wrapped Contexts:
* SPO Occurrences Case Classes:
* Occurrence : (URN, Context, Attribute, Value) : Statement;
* SubjectOccurrence : (URN, SubjectContext, Predicate, Object) : Occurrence;
* PredicateOccurrence : (URN, PredicateContext, Subject, Object) : Occurrence;
* ObjectOccurrence : (URN, ObjectContext, Predicate, Subject) : Occurrence;
* OccurrenceResource : FunctionalResource (wrapper Monad).
* Transforms over OccurrenceResource wrapped Occurrences:
* Mappings Case Classes:
* Mapping : (URN, Occurrence, Kind, Value) : Statement;
* SubjectMapping : (URN, SubjectOccurrence, SubjectKind, Value) : Mapping;
* PredicateMapping : (URN, PredicateOccurrence, PredicateKind, Value) : Mapping;
* ObjectMapping : (URN, ObjectOccurrence, ObjectKind, Value) : Mapping;
* MappingResource : FunctionalResource (wrapper Monad).
* Transforms over MappingResource wrapped Mappings:
* Relationship Case Classes:
* Relationship : (URN, Context, Occurrence, Mapping) Statement;
* SubjectRelationship : (URN, SubjectContext, SubjectOccurrence, SubjectMapping) : Relationship;
* PredicateRelationship : (URN, PredicateContext, PredicateOccurrence, PredicateMapping) : Relationship;
* ObjectRelationship : (URN, ObjectContext, ObjectOccurrence, ObjectMapping) : Relationship;
* RelationshipResource : FunctionalResource (wrapper Monad).
* Transforms over RelationshipResource wrapped Mappings:
* SemanticWebAlignmentTheory: Purpose driven data (assets), contexts (roles), interactions (actors) via semantic tags / labels /facets formal contexts.
* Qi4j / Elmo / Sesame RDF4J Alibaba: RDF Object Models.
* Model (Data):
* Categories / Assets
* Item / Inventory
* Master / Detail
* Facets / Properties
* Persistence: Index / SolrSail (Schema)
* Model Controllers (Contexts):
* Purpose: Purchase
* Roles: Buyerable, Purchaseable (Items)
* Model Properties: Hierarchies, Master / Detail, Facets, Schema, etc.
* Items Order: Intermediate Items (Contexts), Result.
* Model Views (Interactions):
* Session: aPurchase
* Actors: aBuyerable, aPurchaseable (Inventory)
* Model Properties: Hierarchies, Master / Detail, Facets, Schema, etc.
* Roles Order: Intermediate Roles (Interactions), Result.
* FCA Layers:
* Data: Attributes / Values x Types / Instances scaling. Price / Amount Attributes, Product / Item Objects.
* (Object, Attribute);
* Aggregated Measures: (Measure, Object, Attribute);
* (aMeasure, anObject, anAttribute);
* Information: Data across Dimensions Attributes / Values scaling. Time / Date / Store Price / Availability Attributes (Time / Date / Store Product Price Variation). Dimension aggregates axis Attributes / Values.
* (Object, Measures);
* Aggregated Dimensions: (Dimension, Objects, Measures);
* (Time / Date, anObject, (aMeasure: anObject, aDateAttribute));
* Knowledge: Dimensions Information Data Relationships Measures. (Product / Item, Date, Price Availability / Variation rate: Time / Store Product Price Percentage Variation Tendency). Example: Relationships aggregates Objects with same price variation tendency
* (Relationship, Dimension);
* Aggregated Relationships: (Relationship, Dimension, Measure);
* (PriceIncreaseRelationship, (aTimeDimension / Date / Interval, (aMeasure: aProduct, aProductPrice), priceIncreaseAttribute), aMeasure: anObject, anAttribute);
* Order Relationships:
* (PriceOrderingRelationship, (aPriceDimension / Price, (aMeasure: aProduct, aProductPrice), priceOrderAttribute), aMeasure: anObject, anAttribute);
* Cube Statement:
* (Relationship, Dimension, Measure);
* Relationships, Dimensions, Measures hierarchical Cube Traversal.
* Layers
* Data: Attributes / Values x Types / Instances scaling. Price / Amount Attributes, Product / Item Objects.
* Measures Mappings:
* (Object, Attribute);
* Aggregated Measures: (Measure, Object, Attribute);
* (aMeasure, anObject, anAttribute);
* Information: Data across Dimensions. Attributes / Values scaling. Time / Date / Store / Price / Availability Attributes. Attributes Variation. Dimension aggregates axis Objects / Attributes.
* Dimensional Mappings:
* (Measure, Attributes);
* Aggregated Dimensions: (Dimension, Measures, Attributes);
* (aTimeDimension  / Date / Interval, (aMeasure: anObject, anAttribute), anAttribute);
* Knowledge: Aggregates Dimensions Information Data Relationships. Example: Product Price Ordering / Percentage Variation Tendency across Date /  Time / Interval Dimension Measures.
* Relationship Mappings:
* (Dimension, Measures);
* Aggregated Relationships: (Relationship, Dimension, Measures);
* (PriceIncreaseRelationship, (aTimeDimension / Date / Interval, (aMeasure: aProduct, aProductPrice), priceIncreaseAttribute));
* Order Relationships:
* (PriceOrderingRelationship, (aPriceDimension / Price, (aMeasure: aProduct, aProductPrice), priceOrderAttribute));
* Mappings Matching: Relationships stated for matching Dimensions stated for matching Measures. Infer Data, Information, Knowledge  Relationship annotation / result Attributes.
* Cube Statement:
* (Relationship, Dimension, Measure);
* (Dimension, Unit, Measure, Value);
* Relationships, Dimensions, Measures hierarchical Cube Traversal.
* Encodings:
  + Scaling / Events Sourcing. Streams. (parse / populate SAIL to / from models: Sets, etc.).
  + Primes: Attributes Primes Sequence Product.
  + Bitstring: Attributes Bitstring Position Flags.
  + Base N: FCA Scaling by Layers of N Base aggregated Attributes (2, 4, 8, 16, 32, etc. Layers Base). Sub / super Attribute relationship. Dimensions / axis: Radices of multiple factors.
  + Objects / Attributes scaling: Layers Matching (merge contexts).
  + Data Aggregation Context: Metaclass / Class, CSPO / URNResource.
  + Information Alignment Context: Instance / Role, URNResource / Kind.
  + Knowledge Activation Context: Occurrence / Context, Kind / Statement.
  + Ordered Attributes:
  + Data Clustering: Measures (containment / distance).
  + Information Classification: generalizations / specializations (super / sub).
  + Knowledge Relationships (single / married, Friday / Saturday).
  + Order Encoding / Facets. ToDo.
  + Activation Function: Smaller Base / Larger Base Digits (sorted attributes) Aggregation.
* Models
  + Reference Model: FCA / Hashing. DIDs: URNs Resources. Events Sourcing.
  + URNs: URNFactor (ContextFactor, SubjectFactor, PredicateFactor, ObjectFactor);
  + Factors: URNs as CSPO Roles occurrence values product URNFactor (primes / bitstring). FCA Contexts Objects (Statements) / Attributes (URNs Roles Factors: bitstring / primes values hash lattice).
  + Augmentations: Matching / Inferencing. Upper / Matching URNs Role values Factors of Statements / URNs Role Factors.
  + Statement Aggregation Inference: Statement URN Roles Factors of URNs CSPO Factors values. Inferences embedded in Resources URNs Factors product of Statement URN Roles values.
* FCA Contexts:
* Statement (objects) CSPO Attributes:
* (URNFactor (ContextFactor x SubjectKindFactors : class, SubjectFactors : instance, PredicateFactors : attribute, ObjectFactors : value))
* Reified Statements (attributes), SPO Factors x Kinds (objects):
* (StatementFactors : SPO Factors x Provenance (SubjectFactors x SubjectKind, PredicateFactors x PredicateKind, ObjectFactors x ObjectKind))
* Provenance: Entailment. Provenance x reified SPO StatementFactors / source URNs. Statement entails / entailed by Factors product relations: transitive, reflexive, symmetrical (cause / effect, etc.).
* Alignment / Matching: Factors of matching URNs / Statements aggregated by product.
* Graph Statements (materialize / align) URNFactors attributes. Objects: CSPO Factor:
* (URNFactor, URNFactor, URNFactor, URNFactor)
* SubjectKind attributes. Objects: aggregated Subject Factors:
* (Context : super SKs factors, SubjectKind : matching factors / same Predicates Subject factors, Predicate, Object)
* PredicateKind attributes. Objects: aggregated Predicate Factors.
* (Context : super PKs factors, Subject, PredicateKind : matching factors / same Object Predicate factors, Object)
* ObjectKind attributes. Objects: aggregated Object Factors.
* (Context : super OKs factors, Subject, Predicate, Object : matching factors / same Subject Object factors)
* Order / Flow Mappings attributes. Objects: aggregated State Factors.
* (Context, SubjectKind, PredicateKind, ObjectKind)
* Context: State factor (Kinds) previous / next Kinds Mapping Layout. SPO Kinds State (Kinds factors). Super / Sub Context Alignment. Kinds "joins": materialize / align order / flow Statements. Kinds flow: Statement Kinds / SPO States (Contexts). Order: upper / super / sub hierarchies (Kinds Contexts / factors). Joins matching Kinds factors (flows). Provenance state flows.
  + Sets. Data Aggregation.
  + ResourceURNs (Occurrences). Contexts Activation.
  + DOM (Dynamic Object Model Object Graph Mapper): Functional Dataflow. OpenRDF Sesame Elmo / Alibaba. Interactions Alignments.
* Augmentations
  + Aggregation.
  + Activation.
  + Alignment.
* Ontology Matching: FCA / Hashing
  + Data. CSPO.
  + Schema. Kinds.
  + Behavior. Contexts.
* Inferences: FCA / Hashing
  + Data Ontology Matching.
  + Contexts Ontology Matching.
  + Interactions Ontology Matching.
* DDD: Forms Dialogs Protocol: Suggestions (infer Purposes from Gestures: DCI Matching)
* Detectar mediante lo que yo llamo "Gestures" el Propósito de una Interacción de un usuario (persona o servicio: flujos de navegación) en un Contexto dado, según los Datos del "diálogo" en un protocolo que permita inferir y facilitar la intención o el objetivo de la Interacción mediante "Suggestions".
* Detect by means of user "Gestures" (person or service browsing flows) the Purpose of an Interaction, in a given Context, following Data of a of a "dialog" in a protocol such that the Interaction intention or objectives may be inferred and guided by means of "Suggestions".