**Contents: Mision / Vision**

Distributed Knowledge Base. Functional Syndicated Application Framework (Hypermedia: use cases).

Use Cases (problem / solution).

Problems description.

Distributed systems / micro services access to shared data. Shared data consistency. Ontology matching. Integration (EAI / ESB). Introduction of new features / products integrating over existing (linked) data with Semantic capabilities and enhancements.

Integration by augmentation.

Integration by extension.

Declarative Application Design.

**Solution Approach**

Ontology matching (data, schema, behavior alignments).

Hypermedia Dataflow Activation (reactive / event driven knowledge based contents).

Features:

Augmentations (aggregate aligned sources data / schema / behavior enabling interoperation).

Aggregation.

Alignment.

Activation.

Functional "Dialog" Augmentation Semantics.

**RDF Introduction: Graphs, Triples, Quads**

**RDF for Object Graph Representations**

ToDo.

**Models**

Base Model structure / Context layers hierarchies is as follow:

Resource : Functional URI wrapper.

(Context : Resource, Occurrence : Resource, Attribute : Resource, Value : Resource);

(Statement, Occurrence, Attribute, Value);

(Entity, Statement, Occurrence, Attribute);  
(Role, Entity, Statement, Occurrence);  
(Class, Role, Entity, Statement);  
(Flow, Class, Role, Entity);  
(Behavior, Flow, Class, Role);

Models have layers in class / instance roles (except for input layer) and each upper layer aggregates functionally over the previous:

Input Layer (Resource).

Data Layer (Statement instance, Entity class).

Schema Layer (Role instance, Class class).

Behavior Layer (Flow instance, Behavior class).

This Aggregation, together with Alignment and type Activation mechanisms comprehend the Model core Augmentations.

Explain Model, Context, Resource layers / hierarchies (Facets Resources) / Augmentation / Kinds / Roles. ToDo.

Ontology Levels:

Models have “Ontology” Levels. Levels are Layers (of the Model) which are feed into its input Layer with (instance) Statements aggregated from initial input data (Data Level). Schema Level instances feeds the Model input conforming a Session (context / grammars) ontology Level. Then, behavior Level instances feeds the Model input conforming an Interaction (behavior) ontology Level.

Examples: Source, Session, Interaction declarative application protocol use case. ToDo.

Model Facets:

Models have “Facets” which renders the different ways Model data / schema / behavior could be regarded and used for different purposes, from application development to Business Intelligence and Ontology Matching.

Facets are models implemented the same way other models are with Model Resource Contexts and layers and from the same data. Each Facet implements its own Resource URI wrapper (same URIs, ontology matching, provenance of aligned URIs, Facet pivoting). Then, each Facet has its own Model Context Resource hierarchies having Augmentation / Dataflow functors as Model Resource(s) does.

Semantic / Semiotic Facet:

Resource : Functional URI wrapper.

(Context : Resource, Occurrence : Resource, Attribute : Resource, Value : Resource);

(Attributes, Occurrence, Attribute, Value);

(Object, Attributes, Occurrence, Attribute);  
(Concept, Object, Attributes, Occurrence);  
(Sign, Concept, Object, Aytributes);  
(Context, Sign, Concept, Object);  
(Interaction, Context, Sign, Concept);

(Model, Interaction, Context, Sign);

Dimensional Facet:

Resource : Functional URI wrapper.

(Context : Resource, Occurrence : Resource, Attribute : Resource, Value : Resource);

(Properties, Occurrence, Attribute, Value); Data (Properties: distance / facts).

(Value, Properties, Occurrence, Attribute); Info (Properties distance between Occurrence / previous and Occurrence / next).  
(Measure, Value, Properies, Occurrence); Knowledge.  
(Unit, Measure, Value, Properties);  
(Dimension, Unit, Measure, Value);  
(Concept, Dimension, Unit, Measure);

(Model, Concept, Dimension, Unit);

Functional (Model) Facet:

Resource : Functional URI wrapper.

(Context : Resource, Occurrence : Resource, Attribute : Resource, Value : Resource);

(Statement, Occurrence, Attribute, Value);

(Entity, Statement, Occurrence, Attribute);  
(Role, Entity, Statement, Occurrence);  
(Class, Role, Entity, Statement);  
(Flow, Class, Role, Entity);  
(Behavior, Flow, Class, Role);

(Model, Behavior, Flow, Class);

**Meta Resources**

Meta Resources are used by a Model Meta Model for describing models. Some of them are:

URI

Resource

Context / Context

Subject / Occurrence

Predicate / Attribute

Object / Value

Statement

Model

Kind

ContextKind

SubjectKind

PredicateKind

ObjectKind

Message

Template

Augmentation

Transform

Encoding. Addressing (contents, signatures, contexts). Events publish / subscribe. Dynamic subscriptions / bindings. Subscription, reactive Meta Resource(s). Message flow mechanism: from Model to base layers.

Augmentation: Meta Model. Encode Message, Template, Augmentation and Transforms roles Meta Resources. Data, Session, Interaction Levels (Message, Template, Transform, Augmentation statements).

Match Message subjects to Templates, Augmentations and Transforms roles (bound by CK signatures dataflow).

**Meta Model**

Reify Model Layers, Levels and Facets in a Meta Model with Meta Resources. Use Meta Resources class relations for describing models. Meta Resources describe components and roles of Models according a set of relations:

Subject (Resource) / Context (Statement) / Occurrence (CSPO instance) / Role (Kind) / Attribute / Value.

Metaclass (Occurrence) / Class (Context) / Instance (Attributes / Values).

The aim is being able to describe models using models themselves, maybe translating relations to Model Quad Statements.

The same relations could be used to build a Model in which declaratively state model dataflow behavior (reaction to events). A dataflow specification could be described by the following meta resources (roles):

Message (Subject)

Template (Context / domain)

Augmentation (Occurrence, declarative / service Resources: functors)

Transform (Role / range: Kind transform matches). Resulting Message Attribute / Value roles populated.

ToDo.

**Services, URIs, Resource(s) Functional APIs**

ToDo.

**IDs: Addressing / Encoding**

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

**Augmentation**:

Reactive Context Kind (matching signatures) dataflow.

Message - Model - Template (context) - Augmentation (interaction) - Transform (data) - Model - Message.

Implementation API: Node / Container. Services (URIs Context Kind signatures resolution).

Core Services: Activation Augmentation (Naming).

Core Services: Alignment Augmentation (Index).

Core Services: Aggregation Augmentation (Registry).

Core Services: RDF / OWL Backend (endpoint, reasoning, persistence).

Core Services: DIDs Persistence (sync Node state: events sourcing).

Core Services: Protocol (I/O). Node, Session, Intetaction levels. Base Connector Augmentation API. Event driven URIs dialog / prompts protocol adapters.

**Dataflow**

ToDo.

**Implementation**

Languages, Patterns, APIs, Frameworks. (Container, Node, Model, Service, etc.).

**Deployment**

Protocol plugins (Protocol Service) Connectors. Runtime. Core Services. Endpoints. Dataflow.