Business domain translation of problem spaces. SBI Business Integration (WIP draft)

© 2017. Sebastian Samaruga ([ssamarug@gmail.com](mailto:ssamarug@gmail.com))

**I) Abstract**

**1)​ Introduction**

**1.1)​ ​Description**

Current​ needs / problems.

**1.2) Objectives**

**1.3) Proposal**

**2)​ ​Solution**

**2.1)​ ​Leverage​ ​ existing​ ​ solutions**

**2.2)​ ​Architecture**

Logical​ features. Implementation​ ​features. Lambda. Runat.

Aggregation of knowledge layers: facts, concepts, roles / contexts). Examples.

Semantics + Semiotics.

**2.3)​ ​Enhanced​ ​ deployments**

**3) Deployments**

**3.1)​ ​Instantiate Project bundle.**

**3.2)​ ​Select Metamodels / Connectors**

**3.3) Clients. Visualization. Browser.**

**4)​ ​ Implementation**

**4.1)​ ​ ServiceMix / OSGi​ container:​​ Peers / Project Bundles**

**4.1.1)​ ​Apache​ Camel​ custom​ component ('metamodel:' URI prefix)**

Metamodel routes definitions: 'metamodel:[id]:[layer]:[id]' namespaces. Endpoints.

API extension points 'aspects' declarations (class, interface, template implementations).

Metamodel / Connector 'Connection' endpoints: metamodel:jdbc[id], metamodel:rest[id], metamodel:soap[id], etc.

Metamodel Layer endpoints: metamodel:jdbc[id]:[layer], metamodel:rest[id]:[layer], metamodel:soap[id]:[layer], etc.

**4.1.2) ​Metamodel​ OSGi blueprint namespace ('metamodel' tag)**

OSGi Service interface exported by Metamodels.

Instantiates / exports Camel routes / bindings: connector, layer, MetaModel. Broadcast / pattern URIs.

Metamodel routes (layers: connector / sources (bus?) <-> layers <-> Metamodel Protocol Message Dialog).

Aggregation. APIs (extension points) declarations.

**4.1.3) Connector Bundle Project (Kind of Driver) : Connection**

IO Underlying service / datasource.

Provider (service, source) of wrapper Metamodel service instance.

Routes layers: connector / source (bus?) <-> layer <-> MetaModel.

APIs (extension points) between routes implementation (class, interface, template).

Protocol: Source generated events or 'polling' (ie. query for rows, query for svc. args.).

Protocol: Layers routing 'dialog' through activation flows. pub/sub routes flows through layers when lower/higher layer activates lower/higher layer. API customization.

**4.1.4)​ ​Metamodel​ ​Service Instances**

(declarative metamodel namespace, Connectors 'Connection').

Source (Connector 'Connection' producer / consumer) declaration. IO (RDBMS, Service, etc).

Aggregation. APIs (extension points) between routes implementation (class, interface, template).

**4.1.5) Domain Metamodels Instances**

Source: Connector service / source 'Connection' exposes / consumes REST to Connector Client. Connects with other Metamodels (Message IO)

Aggregation enables merge / integration.

Declaratively aggregated / inferred domains (Purpose ontology alignment). Ranks, Topic, Topic Instance.

Templates / API (extension points) for aggregation / merge.

Domain data / behavior schema / instance data encoded in Metamodel layers CRUD / Message IO.

**4.1.6) Connector Client Project**

JCA / JAF/ AOM / DCI HATEOAS (HAL, JSON-LD) Domain Metamodel Browser.

**4.1.7) Application Project Bundle (Maven archetype)**

Metamodel blueprint instances configuration (declaratively: Connectors, APIs, etc).

APIs (extension points): class, interface, template between Metamodels. Routes query / transform (filter events / msgs.).

At least one 'domain' Metamodel to be public.

**5) Metamodels: Declaratively configured Connectors:**

. RDBMSs.

. LDAP / JNDI / JCR.

. Camel (JMS, legacy).

. Service (REST / SOAP).

. Alignment: Identity (classification: class, metaclass / instance, class / occurrence / instance nesting).

. Alignment: Attributes / Links (regression: class, metaclass / instance, class / occurrence / instance nesting).

. Alignment: Contextual sort (clustering: class, metaclass / instance, class / occurrence / instance nesting).

. Streams (Big Data: Spark, MapReduce, etc.).

. BI: OLAP / Mining.

. Lucene.

. TensorFlow (ML for Alignment Metamodels and Big Data. ‘Distance’ calculation models between two aligned resources in a given ‘axis’ or parent class).

. Drools / Flow CEP / JBPM.

. Purpose Metamodel: Task accomplishment services / QA.

. Others.

**6)​ Metamodel​ API**

Class hierarchy.

Layer scopes for Purpose Metamodels:

Purpose Metamodel: Task accomplishment services / QA.

Purpose Metamodel: aggregation ontology, layer scopes (facts, concepts, roles / contexts : IO, dialog state, behavior templates).

Facts layer: (Resource, Resource, Resource);

Concepts layer: (Statement, Kind, Resource);

Behavior Templates layer: (Concept, Concept Kind, Concept Resource);

Purpose Metamodel: Connector populates behavior templates layer from dialog state from previous facts and other Metamodels. IO activates dialog state to / from behavior templates and aggregation augments roles / contexts. IO parses / renders facts aggregated to / from dialog state in respect to behavior templates contexts.

Purpose facts Connector: Connection IO (render hierarchical flows, prompts, confirmations from dialog state relative to current inputs) into facts to / from dialog state in respect to behavior templates.

Templates (aggregated knowledge) determines what to prompt for input and what output facts apply for the current dialog ‘session’. Other Metamodels augments and get augmented from these interactions (retrieve database records, invoke service, alignment ‘interprets’ user input) thus integrating QA into a broader set of integration use cases.

Current input fact in respect to dialog context resolves next behavior template to be populated into dialog context (question / question, question / answer).

Current output fact in respect to dialog context resolves to expected behavior template(s) to be populated into dialog context (answer / question).

Dialog state session mediates between facts and knowledge in question / answer scenarios (hierarchical flows, prompts, confirmations).

**7)​ Functional​​ Metamodel​ API**

Activation (query / match). Monadic transforms.

**8)​​ Dialog Protocol**

Activation. Messages from one route layer may activate higher / lower layers interaction.

**8.1)​ Messages**

Normalized message format which may encode / activate behavior in any route(s) layer.

Routing: (encoding endpoint routes between parent / child and layer neighbor endpoints routing parts):

(Metamodel, metamodel:model:layer:ctx ( Connection, metamodel:model:layer( Connector, metamodel:model )))

Payload:

(Triple ( Triple ( Triple))) : Aggregated IO.

Synchronization and aggregation between 'neighbor' layers and through layers (connector, connection, metamodel).

Messages with payloads for an event in one layer has (activates) correlated events propagating changes in other Metamodel layer arrangements.

**8.2) Routes**

Custom Camel component routes between Connector / Connection, Metamodel layers and Metamodels.

**8.2.1) Connector / Connection routes**

Connector to Metamodel routes. Connector to Connector routes. Connector endpoint.

**8.2.2)​ Metamodel Layer routes**

(Connection?)

Metamodel Layer routes. Layer to Layer routes. Aggregation. Metamodel layers endpoints.

**8.2.3)​ ​Metamodel routes**

Metamodel to Metamodel routes. Metamodel endpoint.

**8.3) APIs**

Classes, interfaces and templates API for configuring route behaviors.

**8.3.1) Connector / Connection route layer API**

Classes, interfaces and templates

**8.3.2)​ Metamodel Layer route layer API**

Classes, interfaces and templates

**8.3.3)​ ​Metamodel route layer API**

Classes, interfaces and templates

**9) Appendix: Monads**

Parameterized​ type​​ M<T>.

Unit​ function​ (T​ ​ ->​ ​ M<T>).

Bind​ function:​ M<T>​ bind T ->​ M<U>​ = M<U>​ (map​ / flatMap:​ bind​ & bind​ function​ argument returns​ a monad, map implemented​ on top of flatMap).

Join: liftM2(list1,​ list2, function).

Filter:​ Predicate.

Sequence:​ Monad<Iterable<T>> sequence(Iterable<Monad<T>>​ monads).

**10) Links. References**