Business domain translation of problem spaces

June 2017. Sebastián Samaruga (ssamarug@gmail.com)

Abstract: The concept revolves around a network of (perhaps existing network's) Peers which interacts in the aims of fulfilling business Objectives (Task flows) for which Peers are selected to perform regarding their Profiles and a given Purpose they and Objectives are proposed to accomplish. A Purpose is a kind of 'abstract goal' to be met. Profiles are the (maybe evolving) capabilities for the resolution of Objectives problem kinds.

Given business domains 'problem spaces' a 'translation' should be made which encompasses given a set of problem Objectives to be solved in one domain (maybe because of Events in that domain) another set of co-requirements in other domains which triggers new Objectives into the flow which shall be accomplished for the global Purpose to be met.

An example: In the healthcare domain an Event: flu diagnose growth above normal limits is to be translated in the financial domain (maybe of a government or institution) as an increase of money amount dedicated to flu prevention or treatment. In the media or advertisement domain the objectives of informing population about prevention may be raised. And in the technology sector the tasks of analyzing and summarizing statistical data for better campaigns must be done.

The technology empowering such endeavor must not be other than that of the graph-oriented featured semantic web paradigm, with tweaks into functional programming concepts and Big Data / Machine Learning inference providers.

Application Protocols:

For most use cases applications needs to be deployed in ways where they are allowed to interact with other applications and produce or consume services.

Although the proposed application mentioned in this document is thought to be implemented in an end to end full stack manner (from presentation through business logic to persistence) the core components meant to be used here are distributed and functional in nature.

Implementations of the framework shall allow for the discovery and publishing of profile driven endpoints.

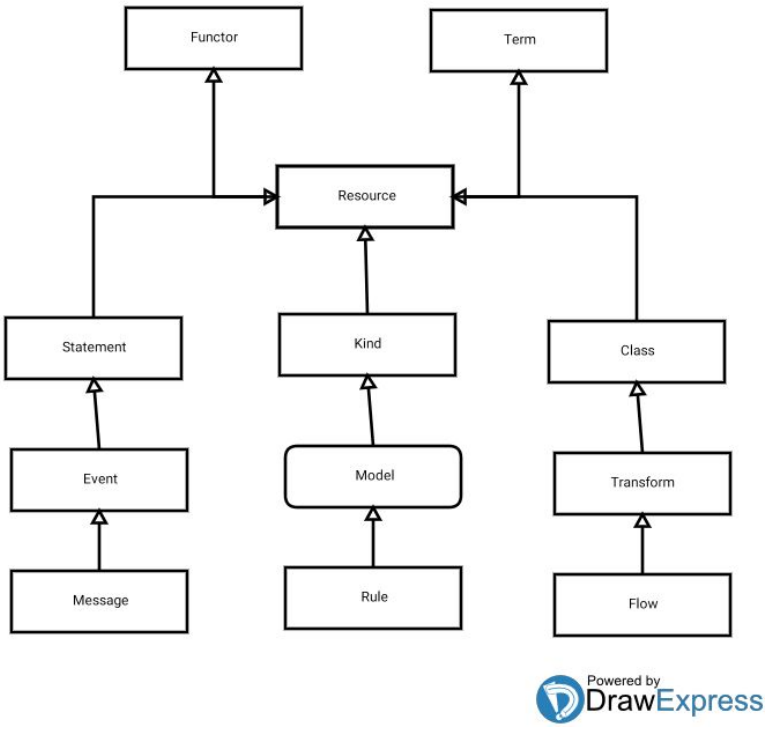
Traditional persistence mechanisms synchronization and standards based protocol bindings (REST, SOAP, others) enables for integration. And a custom schema less dataflow (dialog scoped) protocol is meant to have API bindings in different platforms for application development.

Deployment Peers (Adapter / Port Nodes):

The core architecture deployment is based on Containers of Peer Bindings which allows for Node endpoints Profiles to be discovered / interconnected via the protocols specified by their Mappings / Specs.

This allows for direct (custom semantic dialogs / protocol) Peers integration as also the direct integration of traditional application backends with synchronization capabilities.

Resource Metamodel:



All hierarchy classes (including Functor and Term interfaces) are defined in terms of RDF Quads. Each Quad instance context aggregates other Quads instance contexts according their meaning.

Later we'll see how aggregation combines with dataflow activation for the query/traversal and inference over the resources graph. The Quad statements are of the form:

*) Quad : (Player, Occurrence, Attribute, Value);

*) Term (bound fun / dataflow op) : (ActivateOp, InputPatternActivates, InputPatternConsumes, InputPatternProduces);

*) Functor : (PatternComparison, OccurrenceOp, ContextOp, ValueOp);

Quad patterns:

Resource : (Resource, Resource, Resource, Resource);

Functor : (Functor, Term, Term, Term);

Term : (Term, Functor, Functor, Functor);

Statement : (Statement, Resource, Resource, Resource);

Kind : (Kind, Statement, Resource, Class);

Class : (Class, Kind, Resource, Resource);

Event : (Event, Statement, Statement, Statement);

Model : (Model, Statement, Kind, Class);

Transform : (Transform, Model, Class, Class);

Message : (Message, Model, Model, Model);

Rule : (Rule, Event, Kind, Flow);

Flow : (Flow, Rule, Transform, Transform);

Statements further aggregates:
Facts : Actual input Statements.
Topics : (Topic, Statement, Kinds, Resources);
Purposes : (Purpose, Topics, TopicKinds, TopicResources);

Features (functional programming / providers):

Parsing: Aggregation of basic inputs (Resources, Statements) inferring their Kinds and Classes (a Kind is a Resource Class occurring in an Statement context) allows for further arrangements leading to infer / learn new knowledge. Resource classes are (monadic) parsers of their own types and dataflow activation allows for declarative composition of inputs into new knowledge.

The comparison of resources in respect to a given Term (parent, child, previous, next, current, first, last, single) predicate in a given context (Term, Resource, Context, Resource) is meant to

allow query based inference algorithms and also the (also dataflow enabled) integration of external machine learning engines such as Google TensorFlow (given the correct encoding of resource IDs).

Metaclass - Class - Instance relationships expressed also as order comparison relations (example: a subclass has a superset of the attributes of its superclass, Resources defines other Resources by their aggregation contexts).

Order augmentation: The entities aggregated via Statement hierarchy (Events, Messages) are contextual ordering oriented (temporal context, causal context, etc.) which metadata is used for learning / inference.

Alignment augmentation: The entities aggregated via Kind hierarchy (Models, Rules) are identity resolution / mapping oriented (resolve entities meaning the same thing) which metadata is used for learning / inference.

Attribute / Link augmentation: The entities aggregated via Class hierarchy (Transforms, Flows) are to provide metadata enabling links / attributes augmentation via learning / inference.

Dataflow Activation:

Given a Resource, for example a Message which aggregates three Model(s) that could be: a Facts Model, a Topics Model and a Purposes Model, interaction via dataflow could be a certain Fact (input data Statement) becoming ‘enabled’ (pattern) which ‘enables’ / activates the Message for consumption of the next (info / Topic) Model which then triggers the Purposes Model pattern ‘active’ in Message context.

Triggering and activation of context inputs and outputs (Statements SPOs) is event driven (dataflow) and Resources (Terms) which aggregates its activation states acts as arguments and results of the ‘operations’ defined by their contexts.

Application Stack:

Business Applications (Social Peers Interactions).

Dashboard: ETL / Refine front end.

Application Clients (Platform bindings of Protocol semantics). Functional API (DCI).

BI / Big Data (engines / data sources).

MDM / Governance / ESB / Workflow / BRMS.