Objectives

Dados los mecanismos actuales de manejo de información y de la gestión de los procesos asociados a dicha informacion se pretende proveer a las herramientas actuales de medios aplicativos de la llamada gestión de bases de conocimiento que brinden insights en tiempo real tanto de análisis como de explotación de datos que ayuden a enriquecer con mejoras tanto la utilización del conocimiento como la toma de decisiones.

Description

Un enfoque sería el de “enriquecer” aplicaciones o servicios actuales con conocimiento relacionado al dominio de los mismos en el contexto de una interacción.

El otro podría ser “relevar” servicios y orígenes existentes para integrarlos en el contexto de un despliegue que convenga en exponer toda su funcionalidad actual aumentada y mejorada con servicios de bases de conocimiento y la integración declarativa con otros dominios o procesos.

Business Domain Translation of Problem Spaces

Un ejemplo del segundo punto anterior sería que las instancias de determinados casos de uso en el contexto del dominio de determinada aplicación “disparen” instancias de flujos de casos de uso en aplicaciones de diversos dominios cuya realización está relacionada en algún modo con la realización del primero.

Infer business domain process semantics and operations / behavior from schema and data (and services). Aggregate events, rules, flows.

Architecture: logical view

Peers con Bundles desplegados resuelven y proveen los bindings necesarios (i.e.: persistencia, endpoints) a los recursos configurados declarativamente en los mismos (servicios, nodos, etc.) para que un Port (protocolo) resource pueda proveer a un Binding de plataforma el esquema y los datos necesarios.

Un Bundle puede agregar y mergear diversos orígenes de datos y servicios de aplicación, proveyendo "features" pluggables como descubrimiento e inferencia y el cliente de dicho Binding puede consumir en una interface uniforme los datos, esquema y comportamiento agregados desde los sistemas originales.

Architecture: implementation view

Resource: (Name, Input, Feature, Output); Functional interface.

Service:

- Inputs
- Features
- Outputs

Node: Metamodel / Functional API.

- Inputs
- Features
- Outputs

Peer: Protocol semantics

- Inputs
- Features
- Outputs

Port: Renders CRUD / Behavior flows

- Inputs
- Features
- Outputs

Binding: Client platform endpoints

- Inputs
- Features
- Outputs

Bundle: Declarative Resource Templates

- Internal declarative Metamodel representing deployment Resources. Reactive dataflow activation graph (distributed).
- Services, Nodes, Peers, Ports, Bindings Resource declarations. Subscriptions / transforms.

Node: Functional API / Metamodel. Resources: Metamodel, API activation signature. Node roles: Node Resource implementations: persistence, protocol, IO, alignment, peer, service, aggregation, etc.

Bundles: declarative resources / node bindings (bundle metamodel). Bundle resource implementation role (wraps nodes).

Resource interface IO: message dispatch, routes, endpoints, content type / format / contextual resolution of consumer (from subscription patterns). Protocols (quad / REST HATEOAS). Dialog message augmentation.

Message routing via Resource activation signature (resource, in, feature, out) pattern. Message IO coming from inner / outer Resource layers.

Resource resolution: index, naming, registry. Patterns. Templates.

Resources: Each Resource has its metamodel / functional / dataflow endpoint / interface (templates). Implemented reactive behavior according role (Service: persistence / alignment, Node: merge / augment, etc.): patterns & templates, IO (Resource functional implementation). Declarative Bundle description metamodel: instances & bindings of Resource(s).

Client platform bindings (augment services dialogs via events API over CRUD). Query client contexts over augmented state regarding schema / data of facts, info, knowledge. Common API: standard displaying / protocol (activation). JAX-RS, JAX-WS, JCA.

Example: Persistence Service over Apache Metamodel / JBoss Teiid via D2RQ. Node binding service links federated deployments. Port / Binding Resources expose services through protocol spec + endpoint service (IO).

Metamodel encoding: TensorFlow models. Aggregation. Layers. Reactive / Functional Node API.

Java platform binding: JCA / JavaBeans Activation Framework / XML Beans serialization (DataContentHandlers over standard generic model bean: REST / functional transform verbs over content type). XML / JSON HAL bindings. Export schema for DCI / ORM like bindings.

Bundles deployed as Apache ServiceMix / Red Hat Fuse OSGi bundles.

Domain Use Case

Domain, use cases: Music & Movies retail, record, artist / publisher frontends. Core business cases plus enhancements. Integration with existing APIs.

Features

linkeddata.org / Freebase / DBPedia (async) augmented. Time, places, etc.

Visualización

Visualization: Messages, Resources. Nested (context) tiles. Knowledge interfaces (activation operations).

UX: ZK / ZUL Templates & transforms from endpoints schema metadata / instances (tiles). JCA / JAF / DCI / REST. Activation domain browser.

References

SemanticWeb. OWL. RDF:
http://infomesh.net/2001/swintro/
http://www.linkeddatatools.com/semantic-web-basics
https://www.w3.org/2013/data/

Big Data / Big Linked Data / LDP: https://en.m.wikipedia.org/wiki/Linked_Data_Platform
https://project-hobbit.eu/tag/big-linked-data/

Actor / Role pattern, DCI, Functional programming:
http://www.cs.sjsu.edu/~pearce/oom/patterns/analysis/Actor.htm
https://en.m.wikipedia.org/wiki/Data,_context_and_interaction
https://dzone.com/articles/functional-programming-in-java-8-part-0-motivation
http://www.nurkiewicz.com/2016/06/functor-and-monad-examples-in-plain-java.html?m=1

Reactive programming. Event driven architecture. Dataflow:
https://gist.github.com/staltz/868e7e9bc2a7b8c1f754
https://en.m.wikipedia.org/wiki/Dataflow

ESB / EAI:
Apache ServiceMix.
Red Hat Fuse.

Integración / persistencia:
Apache Metamodel.
JBoss Teiid.

Java platform binding:
JCA (Java Connector Architecture).
JAF (JavaBeans Activation Framework).
Beans Serialization API.
DCI / REST (HATEOAS / HAL).