Dados los mecanismos actuales de manejo de información y de la gestión de los procesos asociados a dicha información 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 / minería de datos que ayuden a enriquecer con mejoras la utilización del conocimiento como la toma de decisiones.

Descripción

Un enfoque sería el de “enriquecer” aplicaciones o servicios existentes con conocimiento relacionado al dominio de los mismos en el contexto de una interacción mediante servicios de conocimiento (endpoints).

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

Business domain translation

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.

La semántica de los procesos y operaciones del dominio de negocios de las aplicaciones integradas se debería poder “inferir” mediante los esquemas y datos de los servicios de dichas aplicaciones.

Posteriormente se podría pretender “agregar” dicho conocimiento en clases e instancias de eventos, reglas y flujos como traducción interna de los modelos del framework del conocimiento del dominio de las aplicaciones integradas para poder reflejar el mismo en deployments subsecuentes.

Características

Orígenes: Desde diversos orígenes de datos, “datasources” y servicios (endpoints REST, SOAP) se pretende “virtualizar” las interacciones con los mismos unificándolos y combinándolos.

Features: Una vez consolidado el “metamodelo” que se obtiene desde los orígenes se pretende proveer los mecanismos para el alineamiento y la aumentación de conocimiento sobre los mismos, ya se mediante agregación, inferencia (de hechos o reglas) u otros métodos que pudieran enriquecer el conocimiento sobre los datos actuales.

API / Salidas: Se pretende proveer de un marco de aplicación unificado (APIs) para la exposición del conocimiento aumentado mediante la utilización de servicios que permitan la interacción con los meta modelos y la sincronización posterior con los recursos que implementen los diferentes tipos de orígenes.

Arquitectura: aspectos lógicos

Resources: Implementación funcional de diversos “backends” para los distintos meta modelos de Nodo.

Nodes: Contienen instancias de metamodelos para diferentes especies de Resource. Proveen API funcional y agregan conocimiento en forma de inferencia sobre sus Resource(s).

Bundles: Arreglo declarativo de Node(s) y Resource(s). Sintaxis de template para especificar rutas y patrones.

Resource types for Node roles, ejemplos: RDBMSResource / DatasourceResource, AlignmentResource, PortResource / EndpointResource (REST / SOAP / etc).

Arquitectura: implementación

Resource: Signature: (Name, Input, Feature, Output); (Resource, occurrence, attribute, value);

Node: Resource kind (node role). Metamodel. Functional interface.

Bundle: Declarative arrangement of Nodes.

Node roles example (by their Resource kind):

Backend / persistence: Inputs, Features, Outputs.

Alignment / augmentation: Inputs, Features, Outputs.

Port / endpoint (CRUD / behavior flows): Inputs, Features, Outputs.

Binding. Client platform endpoints: Inputs, Features, Outputs.

Arquitectura: messaging

Event driven dataflow between Resource, Node and Bundle (export activation signatures). Message I/O.

Events: Signatures / Instances. (resource, input, feature, output). Dimensional aggregated models.

Persistence event example: (datasource, entity, key, object);

Alignment event example, resource instances: (matcher, instance, context, instance);

Endpoint event example: (endpoint, request, state, response);

Events instances only matches corresponding destination signatures.

Caso de uso

Domain, use cases:

Music & Movies (plus DBPedia) 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.

Referencias

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