Semantic Entity Relationship Management

Thorsten H. Niebuhr
A White Paper by WedaCon Informationstechnologien GmbH

1 Abstract

It requires a consistent reorientation and adjustment of current technologies and methods to meet the upcoming challenges of identity management. With this White Paper we would like to introduce the latest development of our Entity Relationship Management system. The system new feature consistently manages and displays all types of entities and their connections to each other based on semantic and ontology approaches.

2 Introduction

Identity and Access Management (IAM) and the 'sister-discipline' Identity Access Governance (IAG) are an integral part of the IT infrastructure in medium and large businesses. These systems manage internal user accounts for employees, system administrators and partners. Increasingly, access rights and accounts of customers and suppliers are considered in an IAM compliant view as well.

This expansion of IAM/IAG application spectrum will increase even further in the coming years. Specifically the emergence of the 'Internet of Things' would make inclusion of “things” into the scope of IAM/IAG system necessary, because these elements often act on the users behalf, or in direct relation with the user.

Today's IAM/IAG systems and processes are mostly not designed to meet those expected and anticipated requirements.

This white paper presents our view on a few identified problems with the current IAM/IAG solutions in section 3 and how we attempt to address them in sections 4 and 5. Sections 6 and 7 present the current status of development and milestones.

3 Limitations of current IAM Systems

Current IAM systems have limitations in respect to a number of functions required for modern and future ready management of digtial identities of any kind. Innovative attempts are necessary to take IAM/IAG to a new level and meet demands of the 21st century.

Human-Machine Communication

Identity management involves defining what users can do on the network and IT systems with specific devices and services, and under what circumstances. Definitions of such access and accounting policies for IAM system processes and workflows performing authorization assignment provisioning is today done using machine optimized polic language. The origin of these policies is however made by the business, in natural language. The policies have to be translated to technical representation requiring close collaboration of the business requesting and technical acting teams.

Policy Management

Additionally the policies are in most cases not centrally managed, but are specific to the programs being integrated with the IAM system. On one hand this is driven by the different demands of the target systems APIs, on the other hand many IAM/IAG solutions offer a fixed set of components exchanging and sharing data in a rather proprietary way and does not allow for flexible control.

Back-end Systems

SQL-Databases and LDAP Directory services are the most commonly used back-end systems for IAM/IAG solutions. Modern, highly scalable and for those purpose optimized back-end systems are rare in the IAM/IAG world and are mostly only available as 'Add-on' (and additional data silo).

Master Data Management

The discipline of master Data Management, the Management of the enterprise data 'offside' user accounts in todays IAM/IAG invironment, is rather exceptional than a rule. Departments, subsidiaries, adresses or other general information like zip-codes and location data usually are managed in external data sheets.

Semantic capabilities

Due to missing Master Data Management information is not used as information in purpose of an semantic approach, but in form of data for date. A time stamp is only used as a number or even a string.

Scalability

IAM/IAG Systems, which have their origin in the early days of IDM usually scale rather poor on behave of the back-end systems not designed for today's requirements. In contrary, modern IAM/IAG systems have an advantage here, but still miss out on required classic enterprise functions. Here we often see a difference of cloud born solutions and those that were offered on the market before scalability and cloud functionality was required.

Entities vs. Identities

Indentities and people (User-Centric IAM) are the primary concern of Identity Management. Other entities (things, divisions, subsidiaries, units, relations) are arranged around the identity (=person). The management of these other entities is usually not consistent with IAM-specific approaches but instead with additional data silo.

Modular vs. Monolith

IAM/IAG- Systems, which present themselves as a 'one piece solution' are in most cases monolithic conceptualized. This complicates the management and leads to system dependencies. Modular performing systems on the other hand perform rather poor with each other for the simple reason that they come along as an add-on and not as a module.

Flexible Adjustments

Perhaps the number one problem in IAM/IAG: The system complexity (and also the continuously growing rulesets they are based upon) complicates a flexible adaptation and the necessary new adjustments, as for example in provisioning or reorganisation of companies.

Authorization and Obligation

The job of an IAM/IAG system is the authorization 'who is authorized to') of access. The control of responsibilities (‘who is in charge (obligation)’) is often overlooked. Failing to respond to one of these basic questions inevitably leads to the necessary use of tools checking compliance and re-certification.

Rapid Deployment

Installation, system maintenance and extensions on existing IAM/IAG- Systems take up significant time. As previously mentioned, the primary reasons are the complexity of prevailing monolithic architecture and poor communication between system modules. Even federal approaches often fail on behalf of the complicated structure of SAML and Co.

Standards

The existing standard’s such as: SCIM, REST, SAML, OAuth2, openID should promote interoperability, however most IAM/IAG provider use proprietary solution. We admit things have increasingly improved in the last years; nevertheless we are still fare away from the aim.

4 Semantic Entity Management

The basic idea of our new approach lies in the possibility of extending processes and requirements of IAM / IAG - systems on all possible entities, and combines it with a semantic structure.

As background, we would like to take a little trip into philosophy: Already the ancient Greeks were meditative the question about concepts of entity, a concrete thing or abstract object: a defined 'being'. The common name of the entity in parlance relates (philosophical) to so many different elements such as things, relations, characteristics, facts or events.

On the other hand, an entity can also represent the ‘nature’ of a thing, an essential property for the existence of the thing itself.

To describe entities in term of their capabilities and their reality, we can use ontologies.

In the recent years of 'semantic Web' development Ontologies have gained importance in computer science. An ontology is a conceptual model of observed reality; a repository of interlinked concept pertaining to a given application domain. Ontologies have outranked the taxonomic approaches that allow only hierarchical classifications.

Coming back to the IAM domain: how could an entity be categorized using ontology in familiar IAM structures?

Lets use countries as example, which are represented by a string in a table column in most IAM systems. In a data model, which uses ontologies there is no table with possible (active) countries used, instead it refers to an Ontology describing countries, which do not necessarily lie in the 'domain' of the organization. This makes a statement such as 'Spain is a country' (subject-predicate - object) possible and turn the date (string) 'Spain' into a concept that has properties and relations making more than a string.

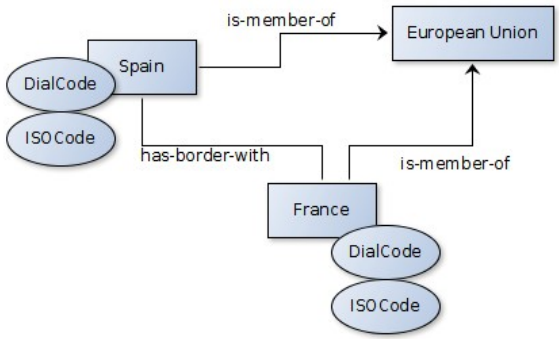


Illustration of a simple ontology: Spain is a european country which shares a border with France. Both are EU-Countries.

When speaking of Ontology we speak of classes of entities and instances of these classes: individual entities. They are related to each other through inheritance and relations defined by properties. Additional 'knowledge' can be added using axioms.

The advantage of ontologies above non semantic data representation lies not only in the presentation of knowledge (their computational usability), but in relations providing richens of contextual description to data.

These relations not necessarily need to be known during the 'design' of ontology; they can be computationally develop from the existing knowledge, and are then available as 'new' knowledge.
Ontology can be represented as a set of 'triples' (subject-predicate-object statements): the same representation is also used in **graph databases**.

Graph databases are todays preferred engines within ‘Social Networks’ to store relations between people. Similarly the relation stored in the graph database are used in e-commerce to display and recommend 'similar' products or additional services of interest based on what were bought by other customers with similar profiles. These technologies offer scalability, what is already presented in these examples.

Graph Databases store 'nodes' (the individual instances, for example 'Spain') as well as their 'edges' (relations to other nodes). Nodes and edges are expanded and further defined by their

properties. A big advantage on graph databases is that these structures are not required to be known in advance, as it is true for databases and directories.

This coincides in high degree with **ontology**, as ontology can be perceived as a graph.
The combination of scalability provided by Graph Databases with ontological models of a semantic make Entity Relationship Management system development possible, allow to repeal a huge number of the limitations of today's IDM Systems.

Furthermore, ontologies make writing policies and rules in 'natural language' possible by using the ontology concepts, which are understood to humans, as constructs, which nevertheless are directly 'machine-readable'.

The IHMC (Institute for Human and Machine Cognition) in Florida, USA has a great reputation on semantically managed System integration, and we are very proud and happy to have them on board for our journey.

Types of policy conflict

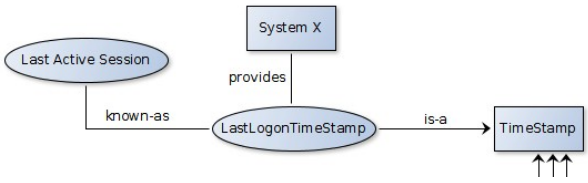
Based on concepts of positive and negative evaluation of authorization (granted / denied) and obligation (required / waived) our system allows to create the relevant policies in constrained natural language. In a case of policy conflict (e.g. required but forbidden; Figure 2), the system automatically tries to resolve the conflict based on predefined algorithms.

> A [**User**] is [**required**] [**to**] [**reset**] his [**Password**] [every **90 days**] .

Example of a policy above uses concepts from ontology (in square brackets).

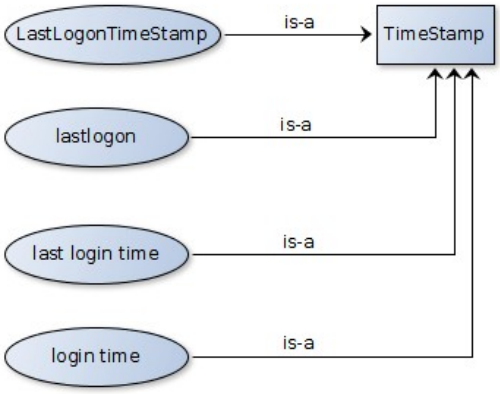
Another example in which ontology shows its strength is in central filing of provisioning rules in and from connected systems, such as schema mappings (Figure 3 and Figure 4).

Let’s have a look on a typical usage scenario in IAM Systems: When did the person last login occur?



Ontology representing concepts related to login

A semantic system does not simply store the timestamp; it stores the fact that it represents a point in time, whatever format is chosen for storage.



Ontology on relations between different forms of timestamp representations

Other useful information that we can derive from this fact and use/proved via ontologies could be

- the Format,
- Conversation Factors,
- origin (Unix, db, ad, LDAP).

5 MicroService Architecture

Modern system architectures and development methods use slim processes; with short, iterative cycles and constant improvement of the implemented elements. This approach has not arrived in the IAM/IAG world yet. Still, large monolithic systems are being developed that in best case are extendable by plugins and add-ons.

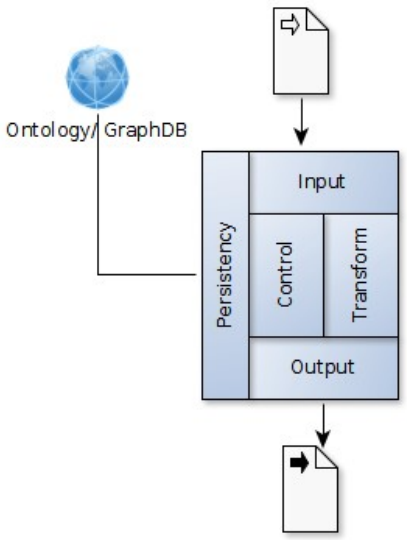
Applications servers, originally thought to be a container for multiple services and applications, are most often suffering from interdependencies between the hosted JAR, WAR or EAR packages or simply share the same system ressources. The common 'solution' to this is to deploy more application servers and distribute the applications on them. Ask yourself: How many application server instances do you have running, and how many applications do they provide per instance?

A modern architecture must above all support one paradigm - Rapid Deployment. This is not limited to usage of Dockers or similiar approaches and methods.

Our Micro Services are an approach/method to IAM/IAG system development that follows the methodology of Rapid Deployment. The main characteristics of MicroService architectures are:

- The ability to easily exchanging single components of the architecture.
- Smart endpoints & (less smart) interpoints.
- Complete redevelopment of components within shortest time (approximately 14 days)
- Fast (automated) Infrastructure
- A MicroService is specialized in performing its roles, and no more. It's purpose determines it's function, not the technology.

Micro Services are also associated with decentralized data storage (persistence) and are dynamically connected to each other.



Micro Service Architecture Example

The Micro Services architecture consists of few layers which have well defined input and output functions to the respective layers.

The first layer (Controller) provides MicroServices self-description and control functions:

- status
- who am I
- who are my neighbors
- what is my task

This layer facilitates the dynamic connection between Services.

The second layer (Transform) is responsible for the actual task the MicroService got assigned. Its the workhorse accepting data streams which are manipulated according to its functions. When the transformation is completed, the resulting data stream is forwarded to the output layer.

The appropriate transformation rules are obtained from the Ontology/GraphDB component for its 'position' and job in the system architecture and performs the transformation of input data based on it.

The persistency of Service state is realized by Persistancer layer (through which communicationbasically takes place).
The transformations at this point also consults the Policies System, in which obligation (required / optional) and authorization (allowed/prohibited) and possible conflicts (required but prohibited) are evaluated and applied to the process data and transformation.

A Micro Service in our environment in first place has a no function 'per se', but gets its 'purpose' directly and dynamically in form of notification by the ontology. Micro Service acting in this environment can operate on more 'complex' or 'smart' features, or only 'simple' activities.

Typical purposes for 'simple', less smart MicroServices

- Transliteration of characters (ü → ue)
- Transformations and conversions
- Regex Tests

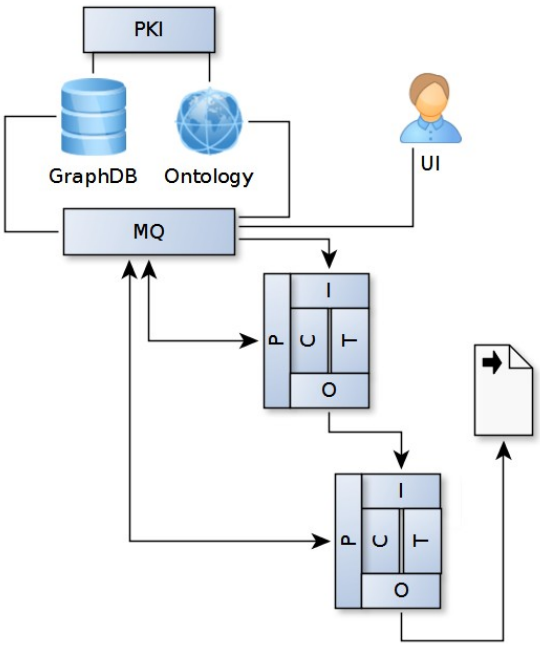
Typical complex (smart) purposes

- Connecting external Applications and API
- Transformation in Standards (SAML Request, OpenID, SCIM, LDAP, etc)

6 Current status

In cooperation with our customers and by partnering with IHMC (Florida Institute for Human and Machine Cognition) we have already successfully implemented and tested a large part of planned components in interaction with our product 'YIAMSuite' during the past months.

YIAMSuite is an IAM solution which consideres relational interactions between Master Data and Identity Data, and already provides intense Entity Management skills since 2011.



Entity Relationship Management System

7 Perspectives

With the solution developed by WedaCon and our partners, as well as with our more than 15 years of experience IAM/IAG range we feel perfectly prepared to take Indentity Management to the next level and essentially influence the area of Entity Relationship Management (ERM).

Contact

WedaCon Informationstechnologien GmbH
Krögerweg 29

D- 48155 Münster

www.wedacon.net
info@wedacon.net