TOPO – Transparent Open Platform for Ontologies

Working Paper · September 2016

DOI: 10.13140/RG.2.2.14365.51687

Duarte Gouveia
Madeira Interactive Technologies Institute
17 PUBLICATIONS **29** CITATIONS
SEE PROFILE

David Aveiro
Universidade da Madeira
66 PUBLICATIONS **320** CITATIONS
SEE PROFILE

TOPO – Transparent Open Platform for Ontologies

Duarte Gouveia[1], David Aveiro[1]

[1] MadeiraInteractive Technologies Institute, Universidade da Madeira, Portugal
duarte.gouveia@m-iti.org, daveiro@uma.pt

Abstract. This paper presents TOPO, a platform to create, change and reuse ontologies, which are conceptual models of the world at a high abstraction level. The goal of TOPO is to mitigate the lack of flexibility of software applications felt by persons and organizations when faced with unforeseen situations or change. Adopting TOPO would allow each information system to grow and adapt to real needs as they occur. TOPO is based on DEMO theory and methodology for (re)design and (re)engineering of organizations. The main contribution of this paper is the presentation of a series of principles and ontological aspects that will be taken in account in TOPO's architecture and implementation. Several innovative and state-of-the-art concepts from computer science, enterprise engineering and neurology are combined in the synthesis of our presented vision.

Keywords: organizational engineering, enterprise engineering, ontology, metadata, data networks

1 Introduction

The belief that it is possible to create a system ontology that is generic enough to be used to model the world, but at the same time concrete enough to enable its automatic transformation into working prototypes, or even full software applications was the reason for creating TOPO. This belief stemmed from the author's professional experience in developing code generation tools that produce fully working database applications from simple entity-relationship models.

TOPO aims at creating, modifying and reusing ontologies, so that the ontologies can be used to solve concrete problems for persons and organizations. We adopt the notion of system ontology as defined in [4], for the "construction and operation of complete systems".

Organizations are ubiquitous and complex social constructions, that can have many forms. We were used to think on organizations as big and well organized structures like businesses, hospitals and factories. Organization theory [2] has evolved to broaden the concept of organizations to include "groups (social structures) whose members coordinate their behavior in order to accomplish shared goals or to put out a product", service or value. With this definition, a group of young people organizing an event through a social network is an organization. Richard Scott [2] argues that organizations can be: rational systems (with collective goals and formalized structure); natural systems (with individual goals but interested in the perpetuation of the organization to achieve them) or open systems (no structure or roles, but flows of interdependent activities with "shifting coalitions of participants").

For many years software engineering has followed the path of building software as a well organized structure. Due to its complexity and stakeholders pressure to change, software development has been shifting toward agile methods that favor gradual growth instead of the “big design up front” approach. This leads to the idea that the de-facto standard pattern of software architecture, “big balls of mud”[3], might be adequate, especially for organizations that are mostly natural systems or open systems, as defined by Richard Scott [2]. Organizations still don't have the software applications that allow them to better handle their natural ambiguity and gradually become better organizations. On the other hand, structured organizations that are based on rational systems, have difficulties in rapidly adapting to change, because their information systems are hard to adapt and evolve, among other reasons.

António Damásio has argued [1] that the reason why our brains are so complex is because we need to handle ambiguity. One might think that the world is objective – that is - a discrete, deterministic and predictable environment and just too big to be handled by our brains or our current computers. We believe that handling subjectivity and ambiguity are central factors in modeling.

The ultimate goal of building applications is to help persons become better, aiding them in deciding, producing, consuming and remembering better. Being humans, we have flaws in our judgments. Most of our daily decisions are based on simple rules that are really just the application of simple models, such as copying what others are choosing as decision criteria. We believe that in the future, people will rely more on software tools to be able to make informed and rational and unbiased decisions.

People and organizations need a new ontology for their software applications, so that they can, for instance, better analyze the current state of the world, predict the future or retrodict (predicting the past).

Ontologies are models of worlds that simplify reality through conceptualization. Since conceptualizations lose information about the world, many different models can be created and all of them can be right and wrong at the same time, depending on each user's purpose. As George E. P. Box said “All models are wrong, but some are useful”. As long as models are able to explain something about the world, they can be useful for some person/purpose, no matter how simple they are. The multi-model approach can broaden understanding of the world and resilience in unexpected scenarios.

Models are complex because they can represent worlds that can be of a continuous or discrete nature. Actions performed on the world can be deterministic or stochastic, that is, have predictable or unpredictable results – especially if the model does not take into account all possible actions that are concurrently happening in the world. Information about the current state of the world might not even be available (or be available but without the required precision). Another factor that contributes to the complexity of models is that the roles of the several actors in the world can change over time from a collaborative mode to an adversarial mode or even be irrelevant to each other.

TOPO aims at developing a system ontology that allows ontologies to be automatically transformed into working prototypes, or even full applications, either by running them as an Adaptive Object Model or by generating code. Over the last 16 years, the author has acquired experience with the development of code generation tools for applications in small enterprises. The author has built several versions of a proprietary code generation tool and used it in its own company to develop Delphi applications (1997-2002) and web based applications (2000-2013) with HTML, MySQL, PHP, javascript and jQuery. From that experience came the need to conceptualize an ontology that would enlarge the domain of applicability of code generation tools from database oriented applications into more generic and powerful software applications. The new ontology should handle subjectivity, flexibility and the need for change. At the same time the new ontology has to appropriately manage the structural business logic, including both the fundamental core ontological aspects and the operational aspects, using the consistent principles uphold by Enterprise Engineering.

2 Foundational Ontologies

The common ground of all ontologies is that there are “things” and “connections” between “things” (boxes and arrows), naming them in several different ways and with different meanings. Apart from these simple concepts, no broad consensual taxonomy of things and connections has been agreed upon. As Admiral Grace Hopper said “The nice thing about standards is that there are so many of them to choose from.”

Foundational ontologies have tried to grasp the fundamental parts of the world, but tend to focus on one or few particular aspects, like: languages, logic and whole-part (mereology) [14]; processes [17], events [18]; logic and semantics [19]. Even the most evolved foundational ontology, DOLCE [20], states that they “do not intend DOLCE as a candidate for a «universal» standard ontology”.

Although these foundational ontologies are substantial philosophical efforts, the majority of their entities lack: a) the ability to be learned with minimal effort; b) the flexibility do adapt to concrete circumstances; c) the beauty of simplicity.

For example, DOLCE taxonomy divides the world in 3 “things” (or entities), in a way that doesn't feel natural: 1) abstracts (fact, set, region, …); 2) perdurants, that is “entities that happen in time” subdivided into events - like achievement or accomplishments - and statives - like states or processes; 3) endurants, that is “things that are in time, and their parts flow with them in time”, subdivided into quality – like temporal, physical or non-physical - and substantial – like physical and non-physical.

Since we don't have a consensual foundational model to guide us, we chose an incremental approach to build TOPO, starting with building simple and broad models and generating code from them. We will later refine TOPO based on experience and iterate, instead of going for the alternative of big design up front.

3 What is TOPO

TOPO stands for Transparent Open Platform for Ontologies.

TOPO aims at being transparent in the sense that it promotes a white-box [4] modeling engineering approach to the development of information systems, that is, a construction combining elements into more complex structures, taking measures [5] to control the combinatorial explosion and the consequent increase of entropy that can arise from that construction as it has to be changed over time [6].

TOPO has the goal of being open in the sense that all elements of its structure are open to be used and improved by the community.

TOPO will be a platform because it aims to provide a set of shared components and a wide range of non-functional requirements to allow the easy construction of a family of applications. Applications generated by TOPO will support social interactions of actors as described in DEMO theory [4].

In TOPO, users and applications will keep the power of initiative to communicate with others whenever they want. TOPO is not a framework since it does not implement the typical inversion of control present in frameworks, by relying on callbacks, and using the Hollywood paradigm [7]: don't call us, we will call you.

4 Implementation Context

Since TOPO aims to being implemented as an information system artifact, some premises must be stated about the environmental context in which it would operate:

- TOPO is the core of a layered architecture composed by applications, agents, interfaces and persons assembled as can be seen in figure 1.

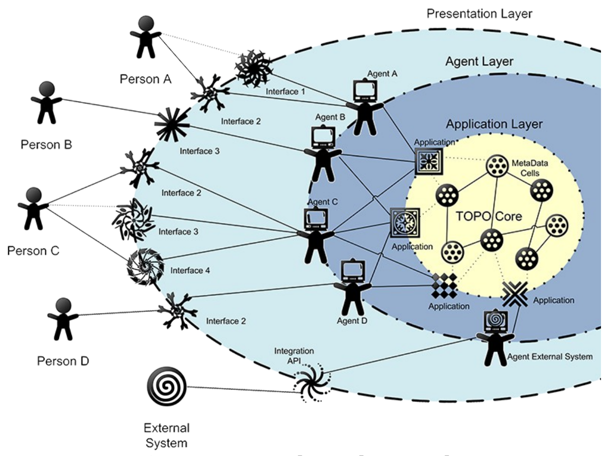

Figure 1 – TOPO layered approach

- Persons can use several interfaces simultaneously to interact with several TOPO applications through an agent.
- Each person has a corresponding agent that represents the person in the system, although agents only perform actions in behalf of persons when previously authorized. The responsibility of actions is always of a person.
- TOPO implementation will be user interface independent and all communications with the interfaces will happen through message exchange.
- TOPO implementation will be in the cloud, although interfaces may keep some cached data.
- TOPO will work over the Internet with multiple devices as view/controller, either browser or mobile device interface (e.g. Android).
- All actors will use message exchange as communication paradigm.
- TOPO will have a maximum response time for each API, otherwise promise a response for later.

- TOPO applications will share common data, but can also have private data for each user/application.
- There will be an integration API for sharing data with external systems. The integration will be performed by a specialized agent.

5 TOPO Structural Options

TOPO is based on two structural options: the usage of concepts instead of primitive data types and having only four types of data, based on the operations they allow (nominal, ordinal, interval and ratio).

5.1 Concepts instead of Primitive Data Types

Programing languages typically have several primitive data types like booleans, characters and several variants for representing numbers. We believe that all these should be modeled as concepts.

For example, strings are sequences of characters and could be handled as sequences of “things”. However, since they are a specific primitive data type, they need specific functions for them. In the PHP programing language there are at least 98 different functions[8] that perform simple tasks only for strings.

Numbers should also be considered as concepts because they have associated meanings that go beyond their digit representation or the number of significant digits. For example, when we see 2014 we see a year, not just a number. Representing all numbers in binary sequences might not even the better way to do it, since numbers do not occur randomly in nature and in society. Benford's law [9], discovered in 1938 and currently used in fraud detection, shows that the occurrence of the first digit “d” in numbers approximates $\log_{10}(1+1/d)$, which gives a probability of ~30.1% of being 1, down to a probability of ~4.6% of being 9. As the mathematician Robert Coveyou said “The generation of random numbers is too important to be left to chance.”

5.2 Only Four Types of Data

In TOPO we chose to represent all kinds of data according to the usual types of data commonly used in statistics: **nominal**, **ordinal**, **interval** and **ratio** (fixed interval with meaningful zero).

Nominal types are the kind of data that do not have numerical properties, even if a number is associated to each category. For example, we can have gender: “male”, “female”, “undefined” [10]. The only operations that make sense to nominal data are checking for equality and counting occurrences. This allows the calculation of mode – as central tendency measure – and the variation ratio, giving some notion of dispersion. In order to manage many nominal types, tools should be provided to handle controlled vocabularies, that is, to allow establishing relationships like broader term, narrower term, use for, use instead and preferred term, as is usual in Information Science.

Ordinal types are constructed over nominal types, adding the condition that they have a function that can specify a total order for all members of a set. However the “distance” between those elements is not a measure, since it only specifies the order for any pair of elements, but not the distance between two elements in the group. Examples of ordinal types are Likert scales in questionnaires. Ordinal types can add median to the available central tendency measures and range, inter-quartile ranges to the dispersion variables. Over ordinal types it is possible to construct ranges (with open or closed boundaries) and operations over ranges and it is also possible to perform sorting operations.

Interval types are constructed over ordinal types, adding the condition that all intervals in the scale are equal. For example, the measurement of temperature in Celsius or Fahrenheit is an interval type, but not ratio (as shown below) because zero degrees does not mean absence of temperature. Intervals allows the usage of meaningful differences, but not the application of ratios between those measures. For interval types it is possible to add Mean as a central tendency measure and standard deviation (and variance) as dispersion measures.

Ratio types are constructed over interval types, adding the condition that there is a absolute zero that is meaningful in the scale and that represents the absence of the phenomenon. Examples are measurements of height, weight and time intervals.

6 TOPO Data Structures

Data structures and algorithms are tightly connected and have a crucial role in software development as they allow efficient solutions to be developed. Algorithms that will handle data-sets over millions of items must have linear or log-linear (N x log N) performance, otherwise they won't be able to complete in acceptable time.

According with the author's experience, the following six data structures have the features to allow efficient representation and support for the majority of algorithms. TOPO will use five usual data structures: **sequences**; **associative maps**; **systems**; **networks**; **finite automata**. All these data structures can be mathematically generalized to **universal algebras**.

6.1 Sequences

A sequence for any finite set of elements can be defined through links that set the order of precedence between them. If the ordering is not complete, that is, not defined for all elements of the set then it will be a partially ordered set.

This simple model can be used to model all kinds of queues, stacks and buffers, since there are only small differences in the way that elements are added and removed from sequences.

6.2 Associative Maps

An associative map is a structure that links a tuple of elements with certain arity (zero, one, two or even more) to another tuple of elements (typically with arity one).

Associative maps can used to represent a vast group of typical structures.

- A **set** instance can be represented as a associative map with arity 1-1 between some possible elements of a domain into the boolean values representing their presence or absence in the set instance.
- A **bag** can be represented as a 1-1 associative map between elements of a domain (without order between them) and natural numbers including zero, for recording their count.
- A **array** maps a natural number to an element. A **matrix** does the same thing in two dimensions, as does a **cube** in three dimensions or a **n-cube** in n dimensions.

- A **function** can also be mapped as an associative array with n-1 arity, linking elements n elements (input parameters), each one of them with a specific domain, to an element of a specific domain. This is the mathematical notion of function where there is a single result that only depended on input parameters influence the result. In TOPO this notion is expanded to include the possibility of returning several results simultaneously or sets of possible results. The mapping between inputs and outputs for a specific function can be done statically or using an algorithm. Functions should log all their calls (time, actor that requested them, input parameters, results, execution time, algorithm used when more than one strategy is available). Logging can be used as a debug tools if errors are located because calls can be backtracked and their implications analyzed. As outputs only depend on inputs, repeated calls with the same parameters can be transformed into a search problem over the log, eventually leading to faster responses. It is also possible to use inverse calls, even when no algorithm is provided to do so. This can be done through the log, or by exhausting all options when there is a limited set of possibilities.

6.3 Systems

Using Jan Dietz definition of system[4], expanded from Mario Bunge, elements can be either members of the composition or members of the environment, and connections are the influence bounds between them. Dietz extended the definition by adding another category of elements, called production, that are the goods or services produced by the composition and delivered to the environment. There should also be added elements called resources that are delivered from the members of the environment to the members of the composition and consumed or transformed into products or services.

The dependencies between the elements of the system are represented using associative maps and sequences.

6.4 Networks

A network (or a graph in mathematical terms) is a data structure that can be represented with four types of elements: **nodes**; **arrows** (directed links between nodes); **edges** (group of directed arrows between the same two nodes with the same direction); and **connections** (group of edges between two specific nodes in any direction). In the nodes and in the links (arrows, edges, connections) between the nodes there can exist associated costs or usage/capacity.

A **tree** is a special kind of network, where each element only has one outgoing arrows, and there are no cycles.

6.5 Finite Automata

A Finite Automata can be used to model state diagrams and grammars for a language. Finite Automatas can be represented as network, where nodes are called states, with two additional bags, containing the states that are initial and the ones that are terminal. For each arrow in the network there might be a constraint that limits the possibility of following that arrow from one state to the other. It is possible to transform between finite automata, regular expression and grammars consistently.

6.6 Universal Algebra

One of the most generic and abstract approaches to generic ontologies is universal algebra – a recent field in mathematics that only started in 1933 by Garret Birkhoff [21]. In universal algebra a world can be described with algebras. An algebra is a set of elements of a universe and connections are a set of operations that take some(arity) elements in the universe and map them to other elements in the universe.

Each operation takes zero (nullary or constant), one (unary), two (binary) or more elements (n-ary) in the universe and transform them into elements that also belong to the universe.

Although these are simple definitions, the elements and the operations can mean anything. The most simple and well known example of algebras are: a) the universe of natural numbers and the binary operations plus and times that take two natural numbers and connect to a natural number. b) the universe with only two members {true,false}, with the unary operation “not”, and the binary operations: “and”, “or”, “imply”.

Elements and operations in a algebra can a) satisfy common features like the arity of the universe or the arguments in the operations; b) satisfy properties in operations like associativity, distributivity, commutativity, neutral element, absorption element, and many others, that can be expressed unequivocally in mathematical terms and can be verified over real data.

By proving (or checking) which properties apply over an algebra it is possible to name them as: finite, trivial, unary, mono-unary, grupoid, group, Abelian group, semigroup, Abelian semigroup, monoid, quasigroup, loop, ring, ring with identity, semilattice, lattice, bounded lattice, boolean algebra, heyting algebra, n-value post algebra, ortholattice, sub algebra, quotient algebra, free algebra, ...

Mathematicians underwent significant efforts in characterizing these structures and establishing rules between then, namely: isomorphism, congruence and homomorphism. "In achieving this, one discovers general concepts, constructions, and results which not only generalize and unify the known special situations, thus leading to an economy of presentation, but, being at a higher level of abstraction, can also be applied to entirely new situations, yielding significant information and giving rise to new directions." [21]

7 Ontological aspects of TOPO

7.1 DEMO methodology in TOPO

DEMO methodology[4] describes the world using transactions. Each transaction can be initiated by a set of roles, but is executed by a specific role. A ontological transaction follows a universal pattern with the sequence of coordination acts request (by requester), promise (by executor), state (by executor) and accept (by requester), complemented with cancellation of previously taken acts in the sequence. The executor of each transaction also performs a production act (execute) between the coordination acts of promise and state. Transactions at certain acts can initiate other transactions and have dependencies on another transactions. These dependencies are specified in action rules, one for each act in each transaction.

In DEMO methodology there is also a State Model where where concepts are connected to other concepts through relationships, and properties are associated to each concept.

TOPO will represent a DEMO ontology using a system. For each act (coordination or production) there is a corresponding action rule. Each action rule is defined as a finite automata where each node corresponds to a task. Tasks can be of several types, that can be either automatic or require human intervention. For example, tasks can include predefined querying some data from the database, flexible querying on the database, showing some data to the user, requesting some input data from the user, validating input data, storing data, requesting data from web service, etc.

For each task that interacts with the user there is a corresponding interface fragment that can be automatically generated based on the required input data, but can also be fully customized to adapt to specific visualization or data input requirements.

Several interface fragments can be combined in a single user interface, depending of the options and constraints of the device being used.

A task and the corresponding interface fragment can be shared by several actions.

From the authors experience with the code generation tool, a group of 125 types of user interface fields have been identified with the corresponding view, edit and search alternatives. In future research they will be synthesized and patterned.

The arrows in the finite automata correspond to possible actions either automatic (like in the case of if, while, foreach control structures) or requiring user decision when explicit decision is required

In TOPO, besides actor roles, there are also users, agents and functions. As presented in section 4, agents are the singleton representatives for users, which can perform action on their behalf when properly authorized to do so. Functions, correspond to organizational roles. TOPO will allow a flexible authorization structure specifying which persons are associated to which functions and which of these can fulfill which actor roles.

Functions also perform a crucial role in the delegation, as persons can delegate withing the organizational hierarchical functions to perform either production acts, coordination acts or both (with or without reporting back to the delegate on performed acts).

Agents perform coordination acts by sending messages to each other through the specified roles. For each role there are queues (implemented with sequences) and those authorized to perform a certain actor role can view and/or consume the messages in the queue.

7.2 Neuron Model

Concepts can be connected with a neuron like structure. In a simple model, a neuron can be represented with a central body called soma, a tree of dendrites that act as input channels and axon as output channel. If we consider each neuron to be a fact, then dendrites could connect to a arbitrary number of who's, what's, where's, etc. Branches in the dendrite tree can have connections that act as suppressors. This analogy is particularly useful because it can model any boolean expression made up of and's, or's and not's, as it has been shown in neural science.

Neuron model is particularly useful for data retrieval within contexts – the small worlds in networks. If we have a set of fully energized neurons as our current context, it could be possible to energize the structural elements connected to those neurons (who, what, why, when, where, how). Then it would be possible to compute how correlated are other neurons by the level of energy they are getting in their dendrites, and realizing not only the similarities, but also the differences between those new neurons with the one in the current context. In turns it would be possible to join some of those new neurons to the current context and repeat the process, as long as desired. This model of retrieval of information is, in simple terms, the way Antonio Damásio [1] proposes as the way memory works in human minds.

7.3 Zachman Framework dimensions

Zachman Framework dimensions are the elements (who, what, why, when, where, how). These elements are very commonly known as they are the basis for the construction of news.

In TOPO, a “who” can be a “person”, an “organizational unit” or a “role”. A “role” is a generalization of a “person” to allow it to be performed by more that one person over time. “Organizational units” can be arranged hierarchically with great flexibility using the “part-of” structural construction. A person with authority to do so, can link a “person” to a “organization unit” (“part-of”) or a “role” (“instance-of”) with certain mandatory start instant and eventually a end instant. It is also possible to link an “organization unit” to a “role” with the operation “instance-of”.

Elements in “what” are the most complex and versatile element in an ontology definition. Here, artifacts are created and structural relationships are established in order to construct systems, data structures, languages and whatever is required to model the intended application.

A “why” is a tree of “reasons”. For each reason we repeat the question “Why?” until nothing else can be said other that a fundamental value that does not need further justification. Therefore, the “why” can be represented as a three where the branches have “purposes” in free text, and each leaf is a “value”.

In the modeled processes of an organization, the values are built in on the construction of the systems, and do not need to be expressed while in operation. However, when unexpected things happen and decisions have to be taken, values can help to choose the right track among different courses of action, according to what the company has set as their core values and strategy, and also according to value conditions or restrictions that influenced process design at the respective organizational change context. [22]

The dimension “when” handles time references (moments and recurring periods) and time ranges. Time is one of the most problematic topics because society does not use a good model to handle time, since we have multidimensional layers with great level of ambiguity and inconsistencies.

The dimension “where” is a controlled vocabulary that can reference to many spaces, either in absolute terms of in relative terms. Unlike time, there are very well established systems for coordinates (Cartesian, polar) and available tools to use these systems, even in mobile equipments. A location can be set using a global coordinate system (absolute) or a local one using objects or earth magnetic field to establish coordinates.

The dimension “how” can have multiple interpretations, either using action rules previously described; or in a functional perspective of social decomposition of activities in textual form.

7.4 Metadata

Metadata is usualy described as data about data, but what is metadata for some people might be raw data on which others would like to build upon. In TOPO all data is metadata.

There are 3 types of metadata: **descriptive** (that tells features about a “thing”); **structural** (that says how “things” are connected to other “things”) and **administrative** (that tells who has rights about each piece of data and how that data was formed (provenance) and how it should be kept – for how long, how and by whom.

Descriptive metadata will be added to any concept in TOPO. Every descriptive metadata is a stored as a fact that links: a) the concept being described; b) the value (that is also a concept); c) a predicate (not mandatory) stating the type of description being made – also a concept; d) a unit of measure that is only required for interval values (not mandatory); e) the error associated with the measurement (not mandatory), only for ratio types of data and by default in the middle of one order or magnitude below the measurement.

The predicates to be used should, as far as possible, be one of the 15 basic types of Dublin Core [12] or one of its extensions, in order to facilitate future semantic integration between ontologies.

The structural metadata is used to establish fundamental relationships between data. A lot of work has been done in identifying the properties of structural relationships between concepts in foundational ontologies, namely in [13] and [14]. Therefore we will just mention some of its elements and rely further analysis to the references and to further work. In general, structural relations are established between individuals, collections and the universal set of values. Some examples of structural relations are: individual part-of individual; individual instance-of universal; individual member-of collection; universal is-a universal (taxonomic inclusion); universal partonomic-inclusion-of universal; collection extension-of universal; collection partonomic-inclusion-of collection; collection partition-of individual.

Unlike descriptive and structural metadata, administrative metadata on TOPO will be mostly automatically collected by the system using default options and current context, although the user can change it later on. Administrative metadata will be stored using neuron model and the Zachmann Framework dimensions and using, as much as possible the semantics of Dublin Core and its extensions, as well as the advancements in provenance of digital documents from information science [15], [16].

TOPO will use the paradigm of no raw data deletion, that is, all raw data is stored with timestamps (administrative metadata) and future changes to data are stored as new facts with newer timestamps. Those facts are never deleted to allow time travel, scenario testing and replay. There are always some time references that create forward inferences and data snapshots of the state of the world, for example, in the start of each day of current and last week, start of each weeks of current and last month, etc.

8 Ontology validation through example

Daniela works at a desk-office in a rent-a-car company. Last week her boss assign her the job of helping in the developing of a replacement software system. The boss choose her since she was the person that best knew the current software and had a global perspective of the company needs. The new software uses a new way of creating software applications called TOPO. The new software development took only one week - which was quite a surprise for everyone in the company – and during that period a software engineer worked in the company with Daniela, training her on how to use the system and navigate in its powerful interface. They were now on the first real work experiments with real customers.

Charles arrives at the desk-office to pickup his rented car. He is an old time regular client that always chooses a low cost category for just a few days, almost all months. Charles makes the reservation through the web site.

The software detects a pattern and notifies Daniela in its interface – the client usually gets the car on Thursdays and returns them on Tuesdays, however, in this reservation the car return was on Monday. TOPO elicited this pattern by calculating how likely was that reservation with those terms (who, when, what, where) and how confident was TOPO based on the amount of records it had to support those claims. Daniela confirms with the client the renting dates, explicitly stating the unusual weekdays.

Charles was surprised that they detected that pattern, and confirms it, and even adds some extra information. He usually takes the 10am flight, but since it was fully booked, he had to pick up the 5am flight, therefore, since the desk-office would be closed at that hour, he would return the car at 8pm in the previous day. This information might be something that is worth registering and eventually considering as possible changes by management, so Daniela adds that unusual information in the system using the flexible nature of the neuron model.

Encouraged by the helpful conversation, Charles asks if it would be possible to rent the car for another day and leave it at the Bellevie Hotel where he is staying. He justifies himself saying that a friend has a birthday party that night and it would be very convenient to have the car...

Daniela checks the information system again, and analyzes the statistics for that client, namely that he never had any problems in any car return and the value the client has for the company. Since her boss has already authorized her to take decisions that improve client satisfaction she decides to try to provide the client with that unusual car return. She calls Hilda, a friend that works at Bellevie Hotels and asks her for keeping the car keys and also a parking space in the hotel garage for Monday night. Hilda says that the garage will be quite full because of a board meeting, but she will reserve a parking space for the client. Hilda also states that she will be working on the hotel reception on Tuesday morning, but that no one from the Hotel will check the car... Daniela accepts the terms and registers those unusual information and tasks in the system, again using a neuron, linking Hilda as a "who", adding a step to the usual car return process, for pick up the car at the hotel with "what", "when", "where" and "why", and assign a role for that task.

Bruno, a colleague of Daniela at the rent-a-car, saw the task and volunteered for performing that task through TOPO. Bruno had a hidden reason for volunteering... He has a crush on Hilda, and since she will be there, it will be an opportunity to meet her.

Tuesday morning Hilda calls Daniela asking for help and stating that they have a problem. The client parked in the garage but hit the hotel's director car. Hilda thinks that the client had a drink too much, because he also overslept, loosing his morning flight. Also, in one hour Hilda has to go to a meeting with the director outside, and then take him to the airport and come back. However the director – cannot move his car now – and Hilda does not have a car...

Daniela realizes she is in trouble and will have to handle the situation carefully. She promises Hilda to be there personally and bring a car to help her and the director. Daniela records those unusual information on TOPO using another neuron.

Bruno sees the new data in the system, and suggests to Daniela to take the limousine as a way to improve the end of the story for the hotel director and Hilda. Daniela searches for similar solutions in the past involving the limousine, finding no precedents, but she agrees with the plan and so they implement it.

Since the client accepted to repair all damages, the hotel director dismissed the incident in the garage and enjoyed the trip in the limousine. Hilda was happy with the solution her friend arranged. Daniela was able to solve everything with the client and give him a ride to the airport. Bruno was able to take Hilda on a date in a limousine and have a chill out moment from that stressful morning.

8.1 Discussion

This illustrative example tries to show that real world and real persons life are much more complex than any model could anticipate. Changing the car return location to the Hotel, without the proper verification procedure would be impossible in a fully standardized company. Solving the problem placed with the car accident with flexibility, handling more concerns than the basic ones, is something that companies would have much difficulty in formalizing in their information system.

TOPO's flexible architecture will allow to record the predictable information of the company processes, but will also allow to add unusual and non standard information's using the neuron model having the several dimensions (who, what, where, when, how, why) linking to specific concepts, either existing or created ones. There are reasons for acting in some ways that would never be part of records in any company information system, like the intentions of Bruno regarding Hilda, or that Hilda and Daniela are friends.

In the rent a car example, the possibility of using the Hotel as a car return location could be dynamically added to the system, as well as a task for an authorization from Daniela, and a task for showing the query of client history in terms of car returns to support that decision, and finally adding two transactions for asking the hotel for that service and another for picking up the car in the hotel by a rent-a-car employee in the following day, that would only be activated if that special case of pickup location after the proper approval. The transaction for the Hotel approval could be implemented through email integrating, assuming that the Hotel does not have a TOPO solution.

The flexibility to change the normal procedure, when done by a responsible and authorized person, can provide a higher level of service quality – through client customization – than what is provided by traditional information systems.

It can be argued that strictly complying to standard procedures are essential for managing. We believe that if its possible to keep situations under control, with records of resources used (what), justifications (why) and persons responsible for those decisions (who), informality can improve quality of service, and address human needs that information systems don't address today.

9 Conclusion

TOPO is in it's initial steps. The authors have chosen an incremental and bottom up approach, so that implementation can take place with a realistic scope and real simple ontologies could be generated into working prototypes.

Many crucial aspects have been left out of this preliminary presentation of TOPO:

- User interfaces;
- Integration interfaces with other systems;
- Governance aspects of organizations;
- Data management on an distributed network;
- Accounting (time, space and flow management in the data network);
- Security aspects like handling faults and controls;
- Performance considerations.

Certainly many constraints and insights will come up from the future steps of specifying TOPO's concepts and implementation aspects taking in account all the dimensions, concepts and innovative ideas presented in this paper. For example, as future work we will explore more in detail how a neuron like structure will be used to integrate the presented dimensions and concepts and how such conceptual neurons will bridge and fuse together the notions of data and metadata giving a quite dynamic quality to both ontologies and systems of people and technology that implement them.

References

[1] Damasio, A. (2012). *Self comes to mind: constructing the conscious brain*. Random House Digital, Inc..

[2] Scott, W. R. (1981). Rational, natural, and open systems.

[3] Foote, B., & Yoder, J. (1997). Big ball of mud. Pattern languages of program design, 4, 654-692.

[4] Dietz, J. L. G. (2006) Enterprise Ontology-Theory and Methodology.

[5] Mannaert, H., Verelst, J. (2009) Normalized Systems: Re-creating Information Technology Based On Laws For Software Evolvability. Koppa

[6] Lehman, M. M. (1996). Laws of software evolution revisited. In Software process technology (pp. 108-124). Springer Berlin Heidelberg.

[7] http://en.wikipedia.org/wiki/Hollywood_principle (last visited January 21, 2014)

[8] http://pt1.php.net/manual/en/book.strings.php (last visited January 21, 2014)

[9] Benford, F. (1938). The law of anomalous numbers. *Proceedings of the American Philosophical Society*, 551-572.

[10]http://www.france24.com/en/20131101-germany-creates-indeterminate-gender-birth-register/
[12] http://dublincore.org/
[13] Bittner, T., Donnelly, M., & Smith, B. (2004, November). Individuals, universals, collections: On the foundational relations of ontology. In *Proceedings of the Third Conference on Formal Ontology in Information Systems* (pp. 37-48).
[14] Guizzardi, G. (2005). *Ontological foundations for structural conceptual models*. CTIT, Centre for Telematics and Information Technology.
[15] Simmhan, Y. L., Plale, B., & Gannon, D. (2005). A survey of data provenance in e-science. *ACM Sigmod Record*, *34*(3), 31-36.
[16] Moreau, L., Freire, J., Futrelle, J., McGrath, R. E., Myers, J., & Paulson, P. (2008). The open provenance model: An overview. In *Provenance and Annotation of Data and Processes* (pp. 323-326). Springer Berlin Heidelberg.
[17] Gruber, A., Westenthaler, R., & Gahleitner, E. (2006). Supporting domain experts in creating formal knowledge models (ontologies). In *Proceedings of I-KNOW* (Vol. 6, pp. 252-260).
[18] Scherp, A., Franz, T., Saathoff, C., & Staab, S. (2009, September). F--a model of events based on the foundational ontology dolce+ DnS ultralight. In*Proceedings of the fifth international conference on Knowledge capture* (pp. 137-144). ACM.
[19] Grau, B. C., Horrocks, I., Kazakov, Y., & Sattler, U. (2007, January). A Logical Framework for Modularity of Ontologies. In *IJCAI* (Vol. 2007, pp. 298-303).
[20] Gangemi, A., Guarino, N., Masolo, C., Oltramari, A., & Schneider, L. (2002). Sweetening ontologies with DOLCE. In *Knowledge engineering and knowledge management: Ontologies and the semantic Web* (pp. 166-181). Springer Berlin Heidelberg.
[21] Birkhoff, G. (1933, October). On the combination of subalgebras. InMathematical Proceedings of the Cambridge Philosophical Society (Vol. 29, No. 04, pp. 441-464). Cambridge University Press.
[22] Burris, S., & Sankappanavar, H. P. (1981). *A course in universal algebra* (Vol. 78). New York: Springer-Verlag.
[23] Pombinho, J., Aveiro, D., & Tribolet, J. (2012). Towards Objective Business Modeling in Enterprise Engineering–Defining Function, Value and Purpose. In*Advances in Enterprise Engineering VI* (pp. 93-107). Springer Berlin Heidelberg.

Agradecimentos à ARDITI - Agência Regional para o Desenvolvimento da Investigação Tecnologia e Inovação através do apoio concedido no âmbito do Projeto M1420 - 09-5369-FSE-000001- Bolsa de Doutoramento