Towards a GOD-theory for organizational engineering: continuously modeling the continuous (re)Generation, Operation and Deletion of the enterprise

David Aveiro
Mathematics and Engineering Department, University of Madeira Campus Universitário da Penentea 9020-105 Funchal, Portugal +351291705283 daveiro@uma.pt

A. Rito Silva
Information Systems and Computer Science Department, Instituto Superior Técnico,Technical University of Lisbon Rua Alves Redol 9, 1000-029 Lisboa, Portugal +351213100287 rito.silva@ist.utl.pt

José Tribolet
Information Systems and Computer Science Department, Instituto Superior Técnico,Technical University of Lisbon Rua Alves Redol 9, 1000-029 Lisboa, Portugal +351213540814 jose.tribolet@ist.utl.pt

ABSTRACT

Much time is lost, in organizations, in the handling of unknown exceptions because organizational models are not current or coherent with reality and there is a lack of concepts and methods in organizational engineering (OE), for a continuous and timely update of models of organizational reality. To address these problems, a renowned methodology for OE – DEMO (Design and Engineering Methodology for Organizations) and its underlying theory are improved and extended enabling a precise and integrated modeling of three aspects that we consider to be part of the function perspective of an organization: (1) viability – specification of vital norms of operation that ensure the viability of the organization, dysfunctions and their causing exceptions, (2) change – specification of the organizational engineering processes responsible for Generation, Operation and Discontinuation of organizational artifacts (OAs) – for example, business rules or organizational actors – in order to solve dysfunctions and (3) architecture – specification of design rules that guide the referred engineering processes, restricting the “shape” of their end result – OAs.

Keywords

organizational engineering, organizational change, organizational self-awareness, function, exception

1. INTRODUCTION

Our initial research efforts had the general purpose of understanding and clarifying what the function perspective of an organization should be. The function concept is normally associated with behavior, activity or operation of an organization or of a certain organizational unit like a marketing or IT department, responsible for the respective function [1]. In [2] we find that the function perspective means looking at a system from the point of view of the using system, in terms of provided functionality, i.e., kinds of behavior that can be caused. We regarded this to be an incomplete use of the term function. As a result of a review that we undertook on how this concept is used in such diverse areas as information systems, biology, sociology and philosophy, we found that, besides the aspects of structure and operation, also central to the function concept is the normative aspect [3], that is, the existence of certain normally expected values – norms – for certain vital properties of a system. In an organization, deviations from such norms imply a state of dysfunction that can possibly compromise its viability.

To better understand presented concepts, let’s consider the scenario of Mario’s pizzeria, taken from [2]. We can define two norms for this pizza business: (1) no more than 2 complains can occur per day and (2) income has to be at least 6540 EUR per month. A possible dysfunction is: for three days in a row between 5 to 8 complains were made per day. This can be a very serious situation because, as a consequence, Mario may lose income needed to acquire enough resources and eventually go bankrupt, closing down the business.

Dysfunctions will have a cause which may be known or unknown. If the cause is known, certain resilience strategies may already exist that can be activated to eliminate or circumvent such deviations [4], [5]. Going back to the pizzeria scenario, it is known that the oven is outdated and, around once every three months, in higher temperatures, the temperature fluctuates and the pizzas may get slightly burned or with some parts not cooked. As a temporary solution, it was found that, during at least one day, temperature should be kept at 200°C, so that it can stay stable enough, although pizzas take a little more time to be ready which can be tolerated. If the cause is unknown we will be in the presence of an unexpected exception. The dysfunction will have to be handled in a way that this concrete cause or unknown exception is discovered and actions are undertaken that either eliminate or circumvent such cause, solving the dysfunction. The first time pizzas were coming out of the oven, half cooked or a little burned it had, as a consequence, the above mentioned dysfunction of complaints. As a result of the handling of this dysfunction the root cause of occasional temperature fluctuations was discovered. The chosen solution of lowering the baking temperature for a day is an example of an implemented resilience strategy that circumvents a cause of dysfunction.
The handling of dysfunctions constitutes another central aspect of the function perspective, namely change through the (re) Generation, Operationalization and Discontinuation of OAs which will eliminate or circumvent the determined cause of dysfunction [5].

Let us now consider that the dysfunction of complaints happens again after some months but no information about diagnosis and solution was kept. Again, one week of monitoring would be needed for a diagnosis and determining a solution. Keeping information of exceptions associated with rules prevents that rules are inadvertently stopped to be used and can save much time.

Summing it up, regarding the function perspective, we conclude that it is a holistic and integrative view of (1) Operation, (2) specification of what is normal operation and (3) change through (re)Generation Operationalization and/or Discontinuation of OAs while handling detected unexpected exceptions. Our proposed notion of function perspective extends the notion of looking at a system from the point of view of the using system simply in behavioral terms – like in [2] – to include the aspects of what is normal behavior and the history of implemented changes in the construction of the system as to solve dysfunctions.

In the remainder of this document, section 2 develops on the problem and motivation underlying our research. Section 3 presents related work that addresses the defined problem. In section 4 fundamental notions of DEMO are reviewed and improved and some new notions proposed, which are at the base our solution proposal. Section 5 presents a proposal of extensions to DEMO which address the defined problem and section 6 has concluding remarks on the solution, its limitations and future work.

2. PROBLEM AND MOTIVATION

Above findings helped us to identify two relevant and closely interrelated more focused problems which are addressed in this paper. On one hand, a large amount of time is lost, in organizations, in the handling of unknown exceptions causing dysfunctions. On another hand, current OE approaches seem to lack in concepts and method for a continuous update of organizational models, so that they are always up to date and available as a more useful input for the process of continuous change of organizational reality and decision on possible evolution choices. We focus on these problems in the context of small timely changes, as opposed to large impact changes in the context of IT/IS projects, mergers, acquisitions and splittings of organizations. Why are these problems relevant? As witnessed in the case of complaints on pizzas caused by temperature fluctuations in the oven, exceptions can seriously compromise the viability of an organization if not managed adequately. Also, exception handling can sometimes take almost half of the total working time, and the handling of, and recovering from, exceptions is expensive [6].

What causes can be identified relating to these problems? We identify what seems to be a lack of capture and management of relevant information of past unknown exceptions and their handling. Many events (which were previously unknown exceptions) can have already been known or expected in the past, but can be (frequently) forgotten and become again unexpected (unknown) due to (1) absence of explicit representation of (i) specific exceptions and actions that were executed (in an Ad hoc and unstructured way) for their handling and (ii) engineered OAs to solve them [7] or (2) removal of human agents from a certain organizational actor role which had established and tacitly memorized specific (informal) rules to handle specific exceptions occurring in such actor role [6].

So it seems that the root problem for the above mentioned interrelated problems is an absence of concepts and method for explicit capture, and management of information of exceptions and their handling, which includes the design and selection of OAs that solve caused dysfunctions. Not immediately capturing this handling and the consequent resulting changes in reality and the model of reality itself, will result that, as time passes, the organization will be less aware of itself than it should be, when facing the need of future change due to other unexpected exceptions.

3. RELATED WORK

In terms of related research, the lack of awareness of organizational reality has been addressed in [8], with the coining of the term “Organizational Self-Awareness” (OSA). This construct has been further refined in [9] and [10]. OSA stresses the importance and need of continuously available, coherent, updated and updateable models of organizational reality. A recently proposed research discipline named Organizational Design and Engineering (ODE) [11], also defends this and further raises the importance of capturing and making organizational history and lessons learned available to organizational actors. OSA, and ODE claim that current OE approaches have the shortcoming of lacking in concepts and methods for a continuous update of models of organizational reality, aligned with the continuous change happening in the real terrain. However, both OSA and ODE have, for the most part, only addressed the issues of identification and formulation of this problem and, in terms of solution, mostly the aspect of representation, leaving the change aspect as future work. This shortcoming of lack of continuous update of models aligned with the continuous change of reality has been addressed, by and large, in research and practice in the context of Workflow Management Systems (WfMS) – see, for example, [7] and [12]. However, current solutions assume that an organization will be using a WfMS, which will not be the case of many organizations. And, even in the case of organizations using WfMS, relevant activities may happen outside of IT context and we may also want to address exceptions related to them.

To ground our solution to the defined problem, we decided to narrow our research focus, choosing a particular OE approach, namely, the Design & Engineering Methodology for Organizations (DEMO) [2]. From several approaches to support OE being proposed, DEMO seems to be one of the most coherent, comprehensive, consistent and concise [2]. It has shown to be useful in a number of applications, from small to large scale organizations – see, for example, [13] and [14] (p. 39). Nevertheless, DEMO suffers from the shortcoming referred above. Namely, DEMO models have been mostly used to devise blueprints to serve as instruments for discussion of broader scale organizational change or development/change of IT systems [14] (p. 58) and does not, yet, provide modeling constructs and a method for a continuous update of its models as reality changes, driven by exceptions.

Our research's contributions – presented in the next sections – include revisions and extensions to DEMO and it's underlying Ψ-theory, with the devising of concepts and a method that systematically address the elicited problem. Before proceeding, the reader which is unfamiliar with DEMO is advised to consult [2] or [13] or other publications in: www.demo.nl.

4. REVISIGN DEMO FOUNDATIONS

4.1 Ontological model as part of the world

We adopt the formal definition of ontological model of a world from [15] as: the specification of its state space and its process space, both expressed in business rules. By state space it is understood the set of allowed or lawful states. It is specified by means of the state base and the existence laws. The state base is
the set of fact types of which instances may exist in a state of the world. The existence laws determine the inclusion or exclusion of the coexistence of facts. By the process space is understood the set of allowed or lawful sequences of events. It is specified by the event base and the occurrence laws. The event base is the set of event types of which instances may occur in the world. Every such instance has a time stamp, which is the occurrence time of the event. From [16] we find that in the Ψ-theory based DEMO methodology, four aspect models of the complete ontological model of an organization are distinguished. The Construction Model (CM) specifies the construction of the organization: the actor roles in the composition and the environment, as well as the transaction types in which they are involved. The Process Model (PM) specifies the state space and the transition space of the coordination world. The State Model (SM) specifies the state space and the transition space of the production world. The Action Model (AM) consists of the action rules that serve as guidelines for the actor roles in the composition of the organization. Central to our research is the SM represented by space diagrams in World Ontology Specification Language (WOSL) [17], [18]. We adopt also the formal definition of world from [15] as: a pair \(<C,B>\) with C being a set of objects, called the composition and B a set of facts, called the state base. The 'existence' of an object is assumed even if one cannot know anything about it. The number of objects in C is denumerable infinite; so, there will never be a shortage of objects: all things that are interesting or that may become interesting are already there.

From the above and, in other words, we can say that the ontological model of an organization can be considered as the set of rules that specifies: (1) the types of facts allowed to exist in the organization world, along with the laws restricting coexistence of facts – i.e., the state space; (2) the types of events allowed to occur in the world, along with laws restricting the sequencing of occurrence of events – i.e., the process space; (3) the guidelines for action, grouped in actor roles, that we suggest to call the action space; and (4) the composition of the organization, in terms of how internal actor roles are composed or decomposed in other (composite or simple) actor roles, which external actor roles exist and how they all interact through transactions, that we suggest to call the construction space. As part of our solution proposal, we argue that all the ontological (or business) rules themselves have to be considered as facts of the organization world. These facts – that we call set OM – describe the current state of the ontological model of the world i.e., the state of the state, process, action and construction spaces of the organization's world. Set OM is, itself, a subset of facts of all the facts that constitute the current state of the organization – called S in [15]. The full history of an organization world is contained in the state base B, the full set of facts of an organization, including current facts (set S) and obsolete facts. The state of the ontological model, naturally, is subject to change. Changes in OM will reflect changes of the construction of the organization, i.e., changes in the state of the several spaces of the organization world. One question that naturally arises as a central issue and is addressed next is: how to represent this subset of facts that constitute the state of the ontological model and how to represent its state changes?

4.2 Ontological representation and world objects model

We adapted the ontological parallelogram from [2] to include the sign concept from the original meaning triangle. This led to a shape change to a trapezium, presented in Figure 1. From now on, we refer to this adaptation as the ontological trapezium.

In [2] it is said that signs are not relevant in ontology, that ontology is about the essence of things, not about how one names them and that designation and denotation become relevant as soon as one wants to communicate. In our view, the claim that signs are not relevant in ontology is contradictory to the fact that ontologies, are born and live in communication. We argue that, not only signs will always be relevant in ontology, as they play a central and fundamental role because they are the means to, through some language, check, share, express and make an ontology explicit, both informally and formally [2]. We argue that one of the purposes of an ontology is, precisely, to communicate, in a clear and objective way, signs that denote the relevant objects from the composition of the organization world and designate the respective relevant subjective concepts so that the intersubjective and objective representations of the organization world are synchronized and coherent. By intersubjective representations we understand the ontological model. By objective representations we understand sets of objective signs in a certain physical medium that can be perceived by human beings. We consider that these objective representations include the diagrams, tables and lists associated with each aspect model of DEMO. We will call the set of all these objective representations the ontological representation.

We consider that the terms fact and object are two ways to call the same objective thing, i.e. a concrete instance of the world, where the former means an intensional notation is used and the latter means an extensional notation is used. An extensional notation means using just an identifier sign that denotes an object. An intensional notation means using a composite sign which is a predicative sentence that, not only denotes an object, but also explains it by means of its constituents. A predicative sentence is a formulation that explains a fact type [17]. The ideas of intensional and extensional notation are taken from the grammar of WOSL [17]. To intensionally denote fact types, we use a notation similar to the evolved DEMO 3 specification [18] which is more friendly and easy to interpret. In [15] a fact is considered to be a particular arrangement of one or more objects. Depending on the number of objects that are involved in a fact, one speaks of unary, binary, ternary, etc., facts. A given example of a unary fact is that Beatrix is the Queen of the Netherlands. Another example of a unary fact is that Beatrix is a human being.

Extending the given example with our proposed notions, HUMAN BEINGS denotes an instance of the world that can be viewed as an object class denoted extensionally by sign HUMAN BEINGS and as a fact type, denoted intensionally by the following composite sign that constitutes a predicative sentence: "<[person] is a human being>". HB01 denotes an instance of the world that can be viewed as an object denoted extensionally by sign "HB01" and as a fact, denoted intensionally by the following composite sign: "<[person: Beatrix] is a human being>".

Since, as a solution to our problem, we want to model change of world objects that constitute set OM and since we also want to do it in a precise and coherent way, we need to precisely specify the rules for generation of world objects, so that we can precisely...
specify rules for their change. We formally define world objects as an arrangement of primitive facts. This arrangement obeys certain rules. The events that characterize the life cycle of a world object also obey certain occurrence rules. All these rules can be specified in what we call the ontological world objects model, considered as the specification of the world objects space. By world objects space we understand the set of allowed primitive facts of objects. It is specified by the object facts base and object facts laws. The object facts base is the set of primitive fact types of which instances may occur in the organization world in order for an object to be defined, i.e., generated. The object facts laws determine the inclusion of the coexistence of primitive facts and of occurrence of events of the life cycle of an object. The definition of the world objects space is similar to the definition of state space of an organization's P-world and, thus, it seems appropriate to also use WOSL to specify it, in what we call: the Organization World Objects Space Diagram (OWOSD), depicted in Figure 2.

![Figure 2: Organization World Objects Space Diagram](image)

The OWOSD presents, in a formal way, the primitive fact types that define objects and facts of an organization world, as well as the primitive result types that characterize the life cycle of a world object. They specify what are mandatory primitive facts, both for an object just with extensional notation and for a fact, i.e., an object with intensional notation – the predicative sentence that explains it. An object has to be generated. It can be target of actions, i.e., an object with intensional notation – the predicative sentence that explains it. An object has to be generated. It can be target of actions, i.e., a sentence that explains it.

4.3 Ontological meta model

In order to model change in an organization world, we need to specify how world objects that constitute set OM are supposed to be arranged to form models, so that we can specify changes in such arrangements which reflect real organizational change. We name all objects constituting OM as model objects. Model objects are arranged in a certain manner to specify the set of allowed models. It is specified by the model base and model laws. The model base is the set of model object types of which instances, called model objects, may occur in the ontological model (set OM) of the world. The model laws determine the inclusion or exclusion of the coexistence of model elements. Just like for the world objects space, the definition of the model space is similar to the definition of state space and, thus, it also seems appropriate to use WOSL to specify the ontological meta-model of a world in what we call: the Model Space Diagram (MMSD).

Just like the set OM was defined as the set of objects of B which are the rules consisting in the ontological model of an organization, it also makes sense that the set of rules consisting in the ontological meta model are also objects of B, constituting another particular set: OMM. Not only it makes sense but it is necessary because, by definition, each world object always has to be an instance of some object class existing in B. So all objects of OM will have to be instances of objects of OMM. One question naturally arises which is: objects of OMM are instance of which classes? This is addressed by our proposal of ontological meta model which, for space reasons, is left out of this document.

Another name for the MSD can be: DEMO Meta Model (DMM), which is the chosen name for the specification provided in [19] and consists in the MSD for the four DEMO aspect models: SM, CM, PM and AM. They are called, respectively: Meta State Model, Meta Construction Model, Meta Process Model and Meta Action Model. We argue that a more precise name for these would be: State Model Space Diagram, Construction Model Space Diagram, Process Model Space Diagram and Action Model Space Diagram. We say this because each of these is specifying, in WOSL, the space of each model, that is, the types of model objects out of which instances can occur in the respective model and coexistence rules governing how to arrange these instances. Each of these models – consisting in model objects, instances of model object types of their respective model space – in turn, specify, respectively, the previously mentioned state space, structure space, process space and action space for B, the state base of the organization.

One major contribution of our research – briefly presented in this section that now finishes – is the revision of some fundamental notions underlying DEMO and the proposal of some new notions, like the ontological representation and the ontological meta model. The other major contribution – presented in the next section – is a set of extensions to the current ontological meta model of DEMO so that we can precisely model change and viability – two aspects of the function perspective of an organization missing in DEMO.

5. BRINGING GOD DYNAMICS TO DEMO

5.1 Viability Model

The focus of our research is change of the organization construction to solve dysfunctions caused by unknown exceptions. Before we focus on how to precisely and coherently model change, we need to focus on how to precisely and coherently model dysfunctions and causing exceptions. As we saw previously, a dysfunction consists in a deviation from a norm of functioning of some particular trait of an organization system. Norms specify values for properties of this system that need to be respected so that via-
bility is maintained. We don't find the aspect of viability addressed in DEMO. One of the contributions of this paper is the proposal of another aspect model to be part of the ontological model, namely, the viability model. For this, we needed to extend the ontological meta model as to include all rules that specify the viability model space. These are present in the Viability MSD in Figure 3 accompanied, next, by an explanation, with examples created for the library scenario [2].

Logically, the viability of an organization will be determined by the specification of norms for properties of production banks, because, inevitably, viability of an organization is directly connected with its production. Properties are a categorical scale whose relevant instances will be created at the ontological model level of a particular organization. Properties will be measured using a particular value unit, also a categorical scale with relevant values defined at the ontological model level. A production bank of the library is PB01, also named membership fee payments. We will need to measure a relevant property of PB01, namely total income per month in euros value unit. The measure fact type is thus explained by the predicative sentence <[property] of [production bank] with [value unit]: [total income per month] of [membership fee payments] with [value unit: EUR]> can be part of the OM of the library as an instance of this fact type.

Object class values will aggregate instances of value units that will need to be referred in other facts. Examples: (V01) <[M01] in [“July 2009”] was [“1340 EUR”]>, (V02) <[M01] in [“August 2009”] was [“845 EUR”]>. The measure fact type allows specification of concrete measurements facts. Examples: (MF01) <[M01] in [“July 2009”] was ["1340 EUR"]>, (MF02) <[M01] in [“August 2009”] was ["845 EUR"]>. The measurement occurrence fact type specifies the frequency with which a certain measurement is done. Examples: (MO01) <[M01] is measured every [“month”]>. A certain transaction type will have to be related with each measure and be responsible for the concrete measurement of reality, with the frequency specified in the respective measurement occurrence, and the production of the respective measurement fact. This will be done with the measures fact type explained by predicative sentence: <[transaction type] measures [measure]>.

Categorical scale VIABILITY STATE allows only two instances <"function"> and <"dysfunction"> to be associated to a measurement fact in another fact, specified by the viability evaluation fact fact type, explained by <measurement fact] evaluated as in [viability state] to [viability rule]>: Examples: (VF01) <[MF01] evaluated as in [function] to [VR01] >, (VF02) <[MF02] evaluated as in [dysfunction] to [VR01] >. The viability evaluation occurrences fact type specifies the frequency with which a certain evaluation is done. Examples: (VEO01) <[M01] is evaluated every [“month”]>. A certain transaction type will have to be related with each measure and be responsible for the concrete evaluation of viability by comparison of the respective measurement fact with the respective viability rule. This evaluation will occur with the frequency specified in the respective viability evaluation occurrences fact, and the production of the respective viability evaluation fact. This will be done with the evaluates fact type explained by predicative sentence: <[transaction type] evaluates [measure]>.

If a dysfunction happens, it can be that it is an occurrence of a dysfunction type caused by an already known exception and for which a certain resilience strategy exists that can solve the dysfunction. A certain transaction type will have to be related with each measure and be responsible for the concrete control of viability by activation of a certain resilience strategy related to the detected dysfunction. Such control will occur with the frequency specified in the respective viability control occurrences fact, and the production of the respective viability control facts. This will be done with the controls fact type explained by predicative sentence: <[transaction type] controls [measure]>.

A resilience strategy normally consists in resuming or suspending action rules, transactions or any other organizational artifact in a way that it will somehow generate conditions that will solve the exception. Let's consider the case that the library regularly sets up courses of history of language, with about 5 alternative classes per week. Classes need to be minimum 70% full to generate the necessary income for expenses. In the case that the starting date of the courses is approaching and registrations are, on average at 40% (dysfunction). There are several resilience alternatives: (1) activation of rule <delay courses start until minimum is reached>; (2) <close classes and call students to transfer them to other classes until minimum is reached>; (3) <start extraordinary advertising campaign, distributing fliers in town center until minimum is reached>.

If current resilience strategies don't function or if it is immediately detected that it is the case of an unknown exception. New organizational artifacts will have to be generated and operationalized to solve this new dysfunction type. This is done through the initiation of an Engineering Transaction, addressed in the following section. But before that, we propose a set of tables to be part

Figure 3: Viability Model Space Diagram
of the VM. As future work, diagrams may be specified to better express the same (or other derived and/or related) information.

We propose a Result types, Properties, Measures and Viability Table (RPMTV) that will show a list of results, their associated properties, measures (including the frequency) and viability rules for such measures. The RPMNV is a clear specification of what is normal operation for relevant results of the operation of an organization. We suggest also a Measures, Observers, Evaluators and Controllers Table (MOECT) that will show a list of all measures along with the transactions and actor roles that have the responsibility of measuring (observing) each measure, evaluating the respective observation facts into function facts and enacting control to solve certain dysfunctions. This may be very helpful in quickly and clearly identifying responsibilities if something goes wrong undetected. Closely associated to the RPMVT and MOECT we propose the Measures and Evaluations Table (MET), which will show, for a certain set of chosen measures and a certain time period, the measurement and evaluation facts produced. This will be essential input for the ones responsible of enacting control and change. A Resiliences Dysfunctions Exceptions and Control Table (RDECT) can specify clearly which resilience strategies exist that can be enabled (resumed) or disabled (suspended) to control dysfunctions on certain properties caused by known exceptions. This can give a general overview of critical OAs (e.g., action rules) that assure viability by avoiding certain exceptions.

5.2 (re)Generation, Operationalization and Discontinuation Model

Change in the state S of the organization, with OM being constant, means organizational operation production change, e.g., generation of result: <book B02 has been published>. Transactions types defined in the OM are responsible for this kind of change. Change in the set OM means organizational construction production change, e.g., generation of result: <transaction type: book reviewing> has been operationalized>. A question that now arises is: how do results that change the state of OM – i.e., create OM elements or change their internal state – come about?

Our solution to this is that OM needs to be “initialized” with a special transaction type that is responsible for creation of result instances of all result types regarding the life cycle of OM objects. We will call this transaction type the engineering transaction type (ETT) and an instance of this type, responsible for state changes in OM, an engineering transaction (ET). A transaction type belonging to OM and which is not the ETT is called an operational transaction type (OTT). Operational transactions (OTs), instances of an OTT are responsible for creation of results that imply a change in elements of B, excluding elements of OM. Result <book B02 has been published> is an example.

We further extend the ontological meta model to include the aspect of change. We propose another aspect model called the (re)Generation, Operationalization and Discontinuation Model (GODM). The GODM Space Diagram, depicted in Figure 4, specifies the model object types out of which instances will exist in OM that characterize the execution of engineering transactions that handle exceptions and change the organization to solve caused dysfunctions.

Following the unknown exception handling cycle from [7], an engineering transaction will execute, in an ad-hoc manner, monitoring and diagnosis acts, with the production of the respective facts and, to remedy dysfunctions, may execute immediate recovery actions, also with the production of the respective facts. We have then three fact types to characterize these three “functions” of exception handling and, thus, we will call instances of these respective fact types as unknown exception handling facts. An ET will also have, as a result, state changes of one or more objects of OM, reflecting real change implemented in the organization to solve or circumvent the exception causing dysfunction. As we saw in section 4.2, the possible types of state changes of a world object can be: generated (creation of a new object or some internal change), explained, operationalized and discontinued. Each of these are modeled in the GODM Space Diagram as a fact type. Instances of these fact types are what we call ontological model change facts.

![Figure 4: Generation, Operationalization and Discontinuation Model Space Diagram](image-url)
GODM. We next fully specify a dysfunction fact with a set of associated facts. We use a written notation in structured english, following a similar style as in the OMG's Semantics of Business Vocabulary and Business Rules (SBVR).^1

**VIABILITY EVALUATION FACT** VEF02: It was a measured fact (MF02) that the measure (M01) of number of occurrences per day (P99) of complains (PB02), in August 23th, was 4, fact that is in dysfunction to the viability rule (VR01) that such measure (M01) conforms to condition "Must always be less than 2" (C01).

This fact will be referred by the ET initiation fact, declaring the beginning of an ET to handle the dysfunction.

**ET INITIATION ETI01:** ENGINEER TRANSACTION ET01 initiated because VIABILITY EVALUATION FACT VEF02 considered to be caused by unknown EXCEPTION E01. ESCALATION FACT EF01: ACTOR ROLE A02 (seller) escalated ENGINEER TRANSACTION ET01 to ACTOR ROLE A03 (boss).

The diagnosis facts of an ET describe the investigation and reasoning process, as well as the determined cause for dysfunction. One of the end results of an ET is that, out of a particular diagnosis fact (conclusion), an exception will be defined and this exception itself is established as the cause of a particular dysfunction kind. We further specify ET01 with the following unknown exception handling facts:

**ET DIAGNOSIS FACT ETDF01:** ET01 diagnosed DIAGNOSIS FACT DF01 (from reading the complains made on August 23th, it seems that the cause may be some problem in the oven, because two complains said the pizzas had raw parts and other two said said the pizzas were hard) – at 4:46 pm, August 26th, 2009

**ET MONITORING FACT ETMF01:** ET01 implemented MONITORING FACT MF01 (during a week, the baker (A02) will measure the oven temperature hourly and pay attention to any unexpected effect in the pizzas) – at 5:06 pm, August 26th, 2009.

**ET DIAGNOSIS FACT ETDF02:** ET01 diagnosed DIAGNOSIS FACT DF02 (the oven is outdated and, according to the baker (A02), around once every week in higher temperatures, the temperature fluctuates and the pizzas may get slightly burned or with some parts not cooked) – at 10:24 am, September 2nd, 2009.

**ET DIAGNOSIS FACT ETDF03:** ET01 diagnosed DIAGNOSIS FACT DF03 (it was found by the baker (A02) that, during at least one day, temperature should be kept at 200º C, so that it can stay stable enough, although pizzas take a little more time to be ready, which can be tolerated) – at 10:27 am, September 2nd, 2009.

ET01 is then fully specified with the consequent ontological model change facts:

**ET GENERATION FACT ETGF01:** ET01 generated, explained and operationalized DYSFUNCTION TYPE DT01 (excessive complains in one day) – at 10:30 am, September 2nd, 2009.

**ET GENERATION FACT ETGF02:** ET01 generated, explained and operationalized DYSFUNCTION OCCURRENCE FACT DOF01 (VEF02 is occurrence of DT01) – at 10:30 am, September 2nd, 2009.

**ET GENERATION FACT ETGF03:** ET01 generated, explained and operationalized EXCEPTION FACT E01 (DF02 is an exception) – at 10:30 am, September 2nd, 2009.

**ET GENERATION FACT ETGF04:** ET01 generated, explained and operationalized DYSFUNCTION CAUSE FACT DCF01 (E01 causes DT01) – at 10:30 am, September 2nd, 2009.

---

^1 available in http://www.omg.org/spec/SBVR/1.0/
6. CONCLUSIONS AND FUTURE WORK

The extensions we propose to DEMO and its underlying theory seem to eliminate its shortcoming, regarding the problems addressed in our research and constitute significant contributions to reduce their impact. We propose an extended notion of the function perspective of the organization, which includes the aspects of viability and change. With our proposal, we are now able to precisely model exceptions causing dysfunctions, the acts and facts of their handling and the facts and acts of engineering transactions that (re)Generate, Operationalize and Discontinue OAs to solve dysfunctions. We, thus, also precisely model changes in the state of the ontological model, reflecting real change happening in the organization world.

The proposed solution of the function perspective and the elicited concept of operation, observation, control and engineer seems to be quite in tune with the well known Plan-Do-Check-Act (PDCA) cycle, of quality management, coined by Shewhart and initially realized in mass scale in Japan tanks to Deming [20]. Engineer would correspond to Plan, Operation to Do, Observation to Check and Control to Act. This is also no surprise since eliminating dysfunction and controlling viability of an organization is all about maintaining the quality of its purpose in relation to its environment. Our proposal of continuously modeling the continuous (re)generation, operation and discontinuation of an organization seems also to be a relevant instrument in the perspective of quality and continuous improvement.

An issue that arises from the formalization of change is that of ownership of Ontological Model (OM) objects in the sense of which actor roles are authorized, for example, to change which rules from which actor roles? Also, it may happen that certain elements of OM should only be available (even in “read only” mode) to certain actor roles. The complex issue of access control and security of the models need to seriously be addressed in future research.

It may appear to be prohibitively complex to model so many facts in the context of Engineering Transactions but it does not need to be so. Although we present facts with a large quantity of information, we do it for readability purposes. With a good combination of a graphical user interface, auto-completion functionality and internal logic, computer technology can ease very much the task of capturing information we’re after, using our solution as a base.

Thus, another line of research will be to develop a method of application of our solution with ICT support so that valuable information to decide on aspects of viability, change and architecture is promptly available to humans in an organization.

7. ACKNOWLEDGMENT

Research work that led to results presented in this paper was possible thanks to the financial support of a PhD scholarship (Ref.: SFRH / BD / 13384 / 2003) subsidized by “Fundação para a Ciência e a Tecnologia - Ministério da Ciência, Tecnologia e Ensino Superior” of the Portuguese government and by the European Social Fund.

8. REFERENCES


