Extending the Design and Engineering Methodology for Organizations with the Generation Operationalization and Discontinuation Organization

David Aveiro[1,2], A. Rito Silva[2,3], and José Tribolet[2,3]

[1] Exact Sciences and Engineering Centre, University of Madeira,
Caminho da Penteada 9000-390 Funchal, Portugal
[2] Center for Organizational Design and Engineering, INESC-INOV
Rua Alves Redol 9, 1000-029 Lisboa, Portugal
[3] Department of Information Systems and Computer Science, Instituto Superior Técnico,
Technical University of Lisbon
{david.aveiro,jose.tribolet,rito.silva}@ist.utl.pt

Abstract. We propose an extension for the Design and Engineering Methodology for Organizations – DEMO – to support organization and model change dynamics: the ontological model of the generic *G.O.D. organization*, considered to exist in every organization and being responsible for the Generation, Operationalization and Discontinuation of organization artifacts – e.g., actor role *pizza deliverer* – as a consequence of the process of handling unexpected exceptions causing dysfunctions in the organization's activity. The G.O.D. organization keeps a thorough trace of all acts regarding the diagnosis of problems (dysfunctions) and the design and operationalization of their respective solutions. Such an historical trace provides useful information to each organizational engineering process (OEP) handling unexpected exceptions. Another benefit is to provide a base for a constantly updated model of organizational reality, useful to guide the general activity of organization agents and to provide up to date information of current organizational reality to each OEP.

Keywords: organizational design, organizational engineering, model, dysfunction, unexpected exception.

1 Introduction

Our initial research efforts had the general purpose of understanding and clarifying what the *function perspective of an organization* should be. Normally, the function concept is associated with behavior, *activity* or operation of an organization or of a certain organizational unit like a marketing or IT department, normally 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 enterprise engineering, information systems, biology, sociology and philosophy, we found that, besides the aspect of *behavior*, also central to

R. Winter, J.L. Zhao, and S. Aier (Eds.): DESRIST 2010, LNCS 6105, pp. 226–241, 2010.

the function concept is the *normative* aspect (e.g., [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*.

We present examples from the scenario of a library, introduced in [2] and extended in our research, as to better accommodate concepts we're proposing. The main activities of the library are *book loaning* and *offer book history courses*. We can define three norms: (1) *min average number of registrants in book history courses 1 week before start is 14*, (2) *min total income per month is 900€* and (3) *max loan declines per week is 30*. A possible dysfunction in the second norm is: *average number of registrants in book history courses is 7 on March 23th 2009*. This can be a very serious situation because, as a consequence, the library may lose income needed to acquire enough resources and eventually go bankrupt, closing down the business.

Dysfunctions will have a *cause* which may be *expected* or *unexpected*. If the cause is expected, certain *resilience strategies* may already exist that can be activated to eliminate or circumvent dysfunctions [4], [3]. Continuing in the library scenario, *it is known that sometimes clients of the library are not enough to fill book history courses and some extraordinary advertising needs to be done, which solves the issue of not enough students to cover expenses*. If the cause is unknown we will be in the presence of an *unexpected exception*. This unexpected exception will have to to be *handled* so that its concrete nature is detected and actions are undertaken that either eliminate or circumvent it, solving the dysfunction. The first time not enough students were registering it had, as a consequence, a dysfunction in the norm of *min total income per month*. As a result of this dysfunction, a handling process was initiated to detect the root cause (unexpected exception) of lack of income, namely: *lack of advertisement of courses*. The resilience strategy *distribute course fliers* was designed and chosen as solution to avoid the referred (previously unexpected) exception.

The handling of unexpected exceptions constitutes another central aspect of the function perspective, namely *change* through the *(re)Generation*, *Operationalization* and *Discontinuation* of organizational artifacts which will eliminate or circumvent the determined cause of dysfunction. We consider an *organization artifact* (OA) as a construct of an organization like a business rule (e.g. "if invoice arrives, check list of expected items") or an actor role (e.g. library member). Change of OAs to handle dysfunctions is considered a special kind of dynamics that – inspired in philosophy literature on this subject – we call *microgenesis* [5]. We find that change is also driven by the detection of *opportunities of improvement* which will increase the viability of an organization and place it ahead of *competition* [6]. This is proactive change, as opposed to reactive change in the cases of resilience and microgenesis.

The focus of our research is on modeling two aspects of reactive change: (1) the resilience dynamics of strategies to solve known exceptions causing dysfunctions and (2) the microgenesis dynamics of handling unexpected exceptions also causing dysfunctions. This paper, in turn, focuses on the second aspect: microgenesis dynamics.

In section 2 we develop our research problem and related work. Section 3 presents our proposed extension to DEMO, in order to express microgenesis dynamics of organizations. Section 4 concludes this paper with a critical review on work done and future lines of research.

2 Problem, Motivation and Related Work

Above findings helped us to identify two relevant and closely interrelated more focused problems. On one hand, *a large amount of time is lost, in organizations, in the handling of unknown exceptions causing dysfunctions*. On another hand, *current Organizational Engineering (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 we saw in the example of the library dysfunctions like lack of enough income can compromise a whole organization. Also, exception handling can sometimes take almost half of the total working time, and the handling of, and recovering from, exceptions is expensive [7].

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 [8] 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 [7].

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.

In terms of related research, the lack of awareness of organizational reality has been addressed in [9], with the coining of the term "Organizational Self-Awareness" (OSA). This construct has been further refined in [10] and [11]. 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) [12], 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, [8] and [13]. 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.

From our review and proposal of a broader notion of the function perspective and related insights brought from Complex Adaptive Systems (CAS) literature, we find that we may have two main types of change dynamics: *resilience* and *microgenesis.* From CAS [4] (p. 33) and philosophy [3], we find that systems maintain an internal model of the world (of themselves and the environment) so that they can activate specific resilience strategies to react, appropriately and in time, to certain known exceptions or fluctuations in critical norms that guarantee the system's viability. We also find that a system adapts with incremental changes [5], having as a main purpose to survive and evolve among competition, by having credit mechanisms which favor changes (adaptations) that increase the system's viability and constitute criteria of measuring success [4] (p. 34), [14] (p. 5). One of the premises from CAS theory is that, to solve new exceptions, "rule pieces" that constitute current resilience strategies that solve similar exceptions may be re-utilized to build new resilience strategies or new organization artifacts to solve the new exceptions. From unexpected exception handling in WfMS [8] and ODE [12], we find that information on the history of organization change is an essential asset in moments where change is again needed, i.e., in microgenesis dynamics.

Modeling resilience and microgenesis dynamics and keeping a systematic history of their execution is deemed as a solution to our main research problem, so that exception handling and organization change is more efficient and effective. Microgenesis is the main focus of our main research project, but to precisely specify its dynamics we needed to precisely specify resilience dynamics. This has been the focus of another report to be published elsewhere. In this paper we will present a brief summary of essential notions proposed for specifying resilience dynamics needed to, in turn, specify the notions of our main focus in this paper: microgenesis dynamics.

To ground our solution, 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, [15] and [16] (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 [16] (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 (microgenesis) nor for the continuous control (resilience) that we need to exert on organizations to guarantee viability.

Contributions of our research – presented in the next sections - extend DEMO, with the devising of concepts and a method that systematically address the elicited problem. While proceeding, the reader which is unfamiliar with DEMO is advised to also consult [2] or [15] or other publications in: www.demo.nl.

3 Applying DEMO to Specify the G.O.D. Organization

One of the main points of the proposed solution in our research is that every organization implicitly has a G.O.D. Organization, responsible for changing the organization system's structure, composition and production. We borrow the ontological system definition from [17] (citing [18]) which concerns the construction and operation of a system. The corresponding type of model is the *white-box model*, which is a direct conceptualization of the ontological system definition presented next. Something is a system if and only if it has the next properties: (1) *composition*: a set of elements of some category (physical, biological, social, chemical etc.); (2) *environment*: a set of elements of the same category, where the composition and the environment are disjoint; (3) *structure*: a set of influencing bonds among the elements in the composition and between these and the elements in the environment; (4) *production*: the elements in the composition produce services that are delivered to the elements in the environment.

DEMO doesn't currently contemplate the aspect of change of the organization system properties above. The "business" of the G.O.D. organization is precisely this kind of change which we have previously labeled as microgenesis. As we also saw previously, microgenesis will be triggered by dysfunctions and implies the execution of organizational engineering processes. After detection of the causing exception, organization artifacts will be generated and operationalized that handle or circumvent such exception, eliminating the caused dysfunction. We will now present our proposal for the ontological model of the G.O.D. organization (GO). We will present it, in a intertwined manner, and in three parts, focusing mostly on the State Model (SM).

In the first part we will focus on the aspect of handling an unexpected exception which will consist in monitoring, diagnose and recovery actions. We propose a table named Monitoring, Diagnosis, Exception and Recovery Table (MDERT) to consolidate information highly useful for this part of microgenesis dynamics.

On the second part we focus on the issue of the life cycle of organization artifacts and on the third we briefly touch on the issues of allocating and training organization agents (human and/or IT based) so that the generated organization artifacts can be operationalized.

3.1 Monitoring, Diagnosis, Exception and Recovery

Following the defined functions of the unexpected exception handling cycle found in [8] and presented in Figure 1, in an OEP, monitoring, diagnosis and recovery acts will be executed in an intertwined and ad-hoc manner, leading to the creation of the respective facts. In [8] this cycle begins with the so called exception detection function.

We argue that the majority of the handling cycle is indeed dedicated to understand and detect (diagnose) the unexpected exception. So it seems exception detection is not the proper name for this first function and that we should use, instead, dysfunction observation, which is the name we give to the first stage of what we call the dysfunction handling process (DHP) – the focus of resilience dynamics in another report of our research – which, in turn, can then initiate an OEP. We find that the area of unexpected exception handling in WfMS – to our knowledge, the most advanced in the characterization of exceptions in the functioning of organizations – seems to suffer from the shortcoming of not clearly separating the notions of dysfunction and exception.

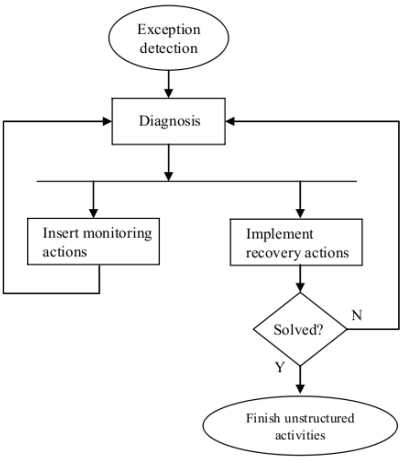

Fig. 1. Exception handling cycle

This shortcoming has also existed in the area of IT Service Management (ITSM), where, with the maturing of standards like ITIL, a clear separation has been done between the concepts of incident and problem, each having a separate process for their management [19]. The processes of incident management and problem management in ITSM have a similar nature, respectively, with our proposals for the DHP and OEP. So one of the contributions of our research is bringing a separation of concerns between dysfunction (incident) handling and unexpected exception (problem) handling to the field of organizational engineering, using DEMO as a base.

Figure 4 presents part 1 of the GO's State Space Diagram (SSD), the formulation of the respective part of the GO's SM. To keep a record of facts resulting from the actions of an OEP, we find object classes, MONITORING, DIAGNOSIS, EXCEPTION KIND and RECOVERY and associated binary fact types and unary result kinds:

- *[monitoring] done in [organizational engineering process]*
- *[monitoring] has been done*
- *[diagnosis] made as a result of [monitoring]*
- *[diagnosis] has been made*
- *[exception kind] detected in [diagnosis]*
- *[exception kind] has been detected*
- *[recovery] done in [organizational engineering process]*
- *[recovery] has been done*

Table 2, that we propose to name as the *Monitoring, Diagnosis, Exception and Recovery Table* (MDERT), presents fact instances of the above mentioned object classes and fact types, for the case of the library.

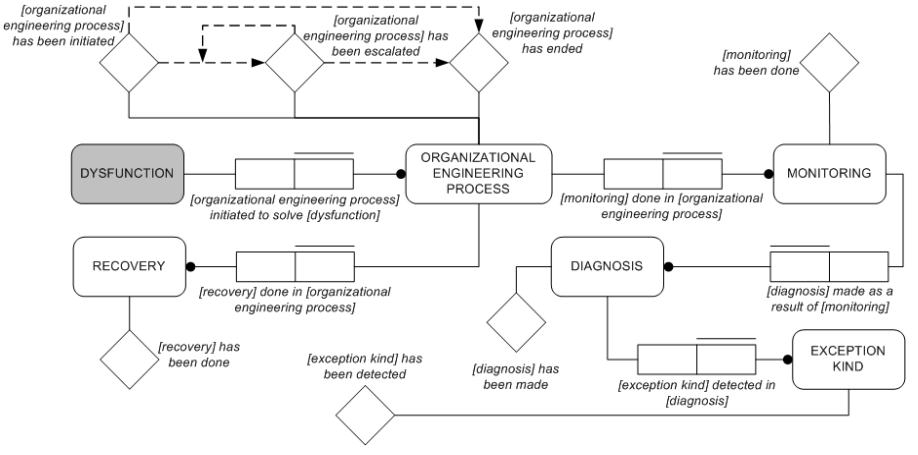

Fig. 2. G.O.D. Organization's SSD – part 1

External object class DYSFUNCTION and fact type: *[organizational engineering process] initiated to solve [dysfunction]* serve as a bridge between the Control Organization – that realizes resilience dynamics – and the G.O.D. Organization. An OEP will necessarily be initiated because of some dysfunction in some viability norm. It may happen that such viability norm was tacit (not formally explicit) but, the moment that it becomes relevant to solve its related dysfunction, it has to be made explicit. This means that the respective organization artifacts (new measure, viability norm, etc.) will need to be generated in the context of such OEP. All OEPs will start with some monitoring action that can have, as a result, some diagnosis action. A diagnosis can, in turn, lead to the detection of a root cause of the dysfunction – an exception kind. In parallel, recovery actions can be immediately executed to eliminate the dysfunction, while monitoring and diagnosis progress.

Table 1. Dysfunctions affecting viability norm VN01 of the Library

viability norm		**measurement**			**dysfunction**	**solved**	**solution**
min total income per month	900 €	total income per month	805 €	Jun 15 2008	DF01	Mar 10 2009	OEP01

Returning to the library scenario, we present, from its Dysfunctions Table the only dysfunction affecting viability norm VN01 – DF01. In the first three lines of Table 2 we find all monitoring and diagnosis facts that led to the detection of the exception kind – *abnormal high rate of loan requests due to exams season* – as causing DF01. We find also a recovery fact, reflecting a recovery action to immediately solve the dysfunction - *increase value of property max_copies_in_loan during exam season*.

Table 2. Monitoring, Diagnosis, Exception and Recovery Table of the Library

handling process	dysfunction	monitoring	diagnosis	exception kind	recovery
OEP01	DF01	verify loan register	loans have been decreasing in the last year	-	-
		ask secretary's opinion on decrease in loan requests	the secretary informed that there are less visitors than usual, but in some months many students have their requests declined due to the max_copies_in_loan limit	-	-
		ask students why they need so many books	every three months its exam season in nearby colleges and they need more books than normal	abnormal high rate of loan requests due to exams season	increase value of property max_copies_in_loan during exam season
OEP02	DF02	do street survey to find cause	most people (70%) didn't saw our add at the newspaper nor the radio, some interested people (10%) suggested to distribute fliers and posters in bars and restaurants nearby	insufficient advertising	delay course start dates for 2 weeks; design, print and distribute fliers announcing courses
OEP03	DF03	do street interview to find cause	great majority of people (90%) show total lack of interest in the course	general lack of interest in courses	delay course start dates until having enough registrants
OEP04	DF06	observe previous resilience and OEP logs	it appears that the general lack of interest in courses is getting worse as delaying courses start is not enough and more drastic actions are needed	general lack of interest in courses	cancel courses and transfer students to prevent dysfunction in monthly income

We next present in Table 3, again from the DFT of the library, dysfunctions affecting viability norm VN02. We find, also in Table 2, actions of OEP02, initiated to solve DF02. While facing dysfunction DF02, the library manager decided to create a questionnaire and request some students to do a street survey to find out the cause for so few applications for the book history course. In parallel he decided, as a recovery action, to delay the start date of the courses for two weeks. As a result of this monitoring action it was found (diagnosed) that *most people (70%) didn't saw our add at the newspaper nor the radio, some interested people (10%) suggested to distribute fliers and posters in bars and restaurants nearby*. The manager then concluded that the exception kind causing DF02 was *insufficient advertising*. Then he immediately proceeded, as other recovery actions, to request to the secretary to design and print course fliers and for her to request some students to distribute them in nearby restaurants and bars. As a result of this OEP some organization artifacts were generated, formalizing the referred recovery actions and constituting a resilience strategy – called *R02 - distribute course fliers*. The topic of generation of organization artifacts is addressed in detail ahead.

Table 3. Dysfunctions affecting viability norm VN02 of the Library

viability norm		measurement			dysfunction	solved	solution
min average # of registrants in book history courses 1 week before start	14	average # of registrants in book history courses 1 week before start	9	Sep 12 2008	DF02	Sep 26 2009	OEP02
			11	Jan 4 2009	DF03	Jan 25 2009	OEP03
			10	Feb 8 2009	DF04	Feb 15 2009	RS02
			8	Mar 23 2009	DF06	Apr 7 2009	OEP04
			7	Apr 18 2009	DF07	May 14 2009	RS04

Regarding OEP03, it was initiated to solve DF03 where resilience strategy RS02 did not stop violation of viability norm VN01. So the manager decided to create another questionnaire to determine the cause while, again, delaying start date of the courses as a recovery action. It was found that, contrary to some months ago, the *great majority of people (90%) show total lack of interest in the course*. This determined the causing exception kind as being *general lack of interest in courses*. It was decided, as a recovery action, to keep delaying courses start dates until having enough registrants. As ahead will be seen, this action gave origin to resilience strategy *RS03 - delay courses start*.

OEP03 was initiated to solve DF03 where the manager, by observing previous resilience and OEP logs, found that neither resilience strategy RS02 nor RS03 could solve DF03 and concluded that lack of interest was simply growing and the best option would be cancel part of the courses and transfer students from the closed courses to the ones still planned to happen. In this manner the minimum average number of registrants would be achieved and a dysfunction in monthly income would be prevented as room renting expenses would not be covered with so few students.

As it should be clear by now, keeping a record of past monitoring, diagnosis and recovery facts aids the creative and ad-hoc process of handling current unexpected exceptions affecting the same viability norm. In our example, we saw how the idea of a street survey was reused from OEP02 to OEP03. Thus, similar (or identical) monitoring and recovery actions may be effectively (re)used in the process of handling a certain exception and we also find that recovery actions may become approved as formal organization artifacts. Namely, recovery actions can become control transaction kinds – to handle the occasional, now known, exception – or normal transaction kinds – to handle this exception which in fact will, in the future, become a frequent and expected event which should become part of normal operation of the organization.

3.2 Organization Artifacts Life Cycle

Figure 3 presents the Meta Construction Model, part of DEMO's Meta Model available in [20].

In Figure 4 we find part 2 of the GO's SM where we see, as a central piece, the ORGANIZATION ARTIFACT object class. This class aggregates all “real” organization artifacts that constitute an organization system's composition, structure and production. Following the ontological parallelogram from [21], one can say that objective DEMO representations – like diagrams – designate subjective (in the mind) concepts

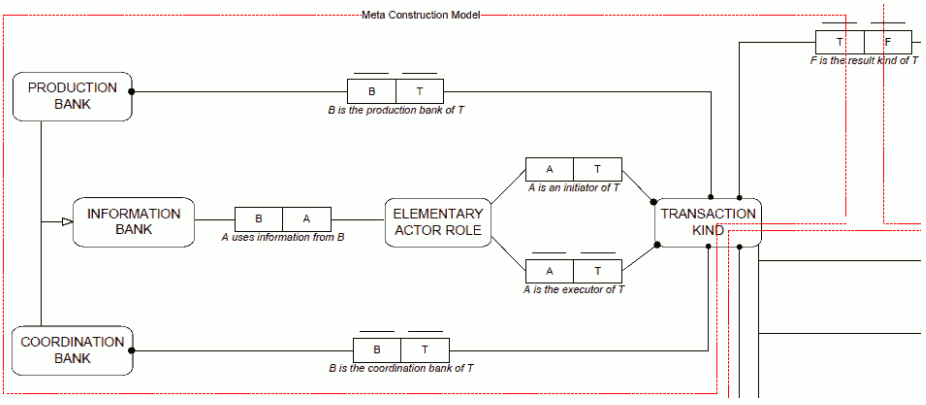

Fig. 3. DEMO Meta Model – Meta Construction Model

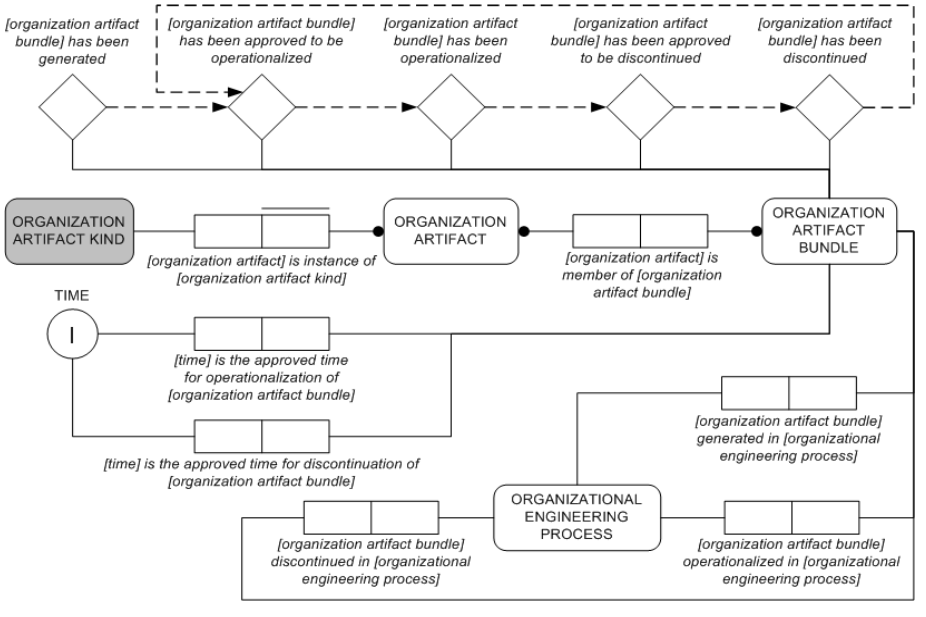

Fig. 4. G.O.D. Organization 's SSD – part 2

that constitute the ontological model (e.g. concept of transaction kind distribute fliers). These subjective concepts will, in turn, refer to “real” objective organization artifacts. They are real because they belong to the collectively shared inter-subjective reality of the organization system but are abstract, in the sense that they have no physical existence. Thus, instances members of object class ORGANIZATION ARTIFACT are referred by their respective concepts of the ontological model or, in other words, the ontological model of an organization is the conceptualization of its set of organization artifacts. Such concepts must be instances of a certain type which is a

generic subjective concept that refers to a class of organization artifacts. Object class ORGANIZATION ARTIFACT KIND, thus, represents all (meta) fact types of the meta model out of which instances can occur in an organization. Because an organization artifact can be both viewed from an objective perspective – as belonging to the shared inter-subjective reality of the organizational self – and from a subjective perspective – as being a concept in the minds of an organization's human agents it seems to be appropriate for the G.O.D Organization to have, in its SM, the fact type: *[organization artifact] is instance of [organization artifact kind]*. That is, the concept of a particular organization artifact will be an instance of a particular type corresponding to a particular organization artifact kind of the organization space. As the GO is where the organizational self is produced, it is essential to keep information of which type a particular organization artifact is, so that, for example, automatic creation of an ontological representation is possible.

Typically, organization artifacts will always be generated and operationalized in bundles. For example, if we add a transaction kind, we necessarily need to add facts specifying which elementary actor role can initiate it, which can execute it, what's the associated result kind, etc. Thus, the need of object class ORGANIZATION ARTIFACT BUNDLE and fact type: *[organization artifact] is member of [organization artifact bundle]*. So, as a result of an OEP, a certain bundle of organization artifacts will be generated which then needs to be operationalized. This is captured by fact types: *[organization artifact bundle] generated in [organizational engineering process]* and *[organization artifact bundle] operationalized in [organizational engineering process]*. It may happen that a certain set of organization artifacts may become obsolete as a result of the soon to be operationalized ones. So it should be formally explicit which organization artifacts are discontinued in a certain OEP. This is possible by creation of instances of fact type: *[organization artifact bundle] discontinued in [organizational engineering process]*. Going back to our example of the library, a series of organization artifacts were generated as part of bundle OAB01. The GO will have to explicitly create instances of fact type *[organization artifact] is instance of [organization artifact kind]* relating each organization artifact with their respective kind and also instances of fact type *[organization artifact] is member of [organization artifact bundle]*, relating the same organization artifacts with bundle OAB01. Several kinds of tables can be specified to express relevant information of organization artifacts. Table 4 presents an example – a proposal of an Organization Artifacts Table (OAT) of the library.

Before being operationalized all organization artifacts of a bundle need to be approved for operationalization. The same applies for discontinuation. The time instant when the approval occurs may be different of the time instant of consummation of operationalization (and/or discontinuation). For example, the approval, by the library administration board, of OAB01, may occur in one day and the operationalization only after some days, due to time needed for implementation. So the time instant for operationalization and discontinuation of organization artifact bundles should be formalized by instances of the following fact types: *[time] is the approved time for operationalization of [organization artifact bundle]*; *[time] is the approved time for discontinuation of [organization artifact bundle]*. Note that in the context of an OEP we may have generated organization artifacts that were never operationalized but can provide good ideas to be reused in the generation of other organization artifacts in future OEPs, or even operationalized in their original shape.

Table 4. Organization Artifacts Table of the library

organi-zation artifact	kind	id	name/predicative sentence	bundle	last state
OA41	exception kind	E01	abnormal high rate of loan requests due to exams season	OAB01	operationalized
OA42	measure	M01	loan declines per week	OAB01	operationalized
OA43	viability norm	VN01	max loan declines per week	OAB01	operationalized
OA44	measure restriction	MRF01	VN01 restricts M01	OAB01	operationalized
OA45	exception causing dysfunction	EDF01	E01 causes dysfunction in VN03	OAB01	operationalized
OA46	transaction kind	T21	general management	OAB01	operationalized
OA47	actor role	A21	director	OAB01	operationalized
OA48	transaction initiation	TIF20	A21 is an initiator of T21	OAB01	operationalized
OA49	transaction execution	TEF20	A21 is the executor of T21	OAB01	operationalized
OA50	viability norm control	VNCF01	T21 controls VN01	OAB01	operationalized

We find, also in Figure 4, all result kinds characterizing the life cycle of organization artifacts. The ones regarding approval serve the purpose to have formal ownership of the relevant parts of an organization system. Discontinuation is an often neglected step regarding organization artifacts, which can lead to effects such as unneeded bureaucracy. Taking again the example of the library, considering that all books get equipped with hidden localizing chips one can eliminate the need of restricting maximum books a member can have on loan since at all times one can always pinpoint with GPS where a certain book is. So the above generated bundle can be discontinued.

3.3 Organization Artifacts Operationalization

Another issue currently not explicitly addressed in DEMO is that of allocation of real agents, for certain time periods, to fulfill certain actor roles. It seems the G.O.D. Organization is the right place to keep information bridging the world of the organizational self with the world of agents and their training and allocation to the respective actor roles as, without this, the operationalization of new organization artifacts cannot be realized. In other words, so that a certain dysfunction caused by an unexpected exception is solved, it is not enough to generate organization artifacts in an OEP. One has to operationalize them and this operationalization should occur in the context of the same OEP. Thus, in Figure 5 we present the SSD consisting in the formulation of the third and final part of our proposal for the GO's SM.

We find object class AGENT TRAINING where instances of this class will be part of a certain OEP and be related with a certain organization artifact bundle to be operationalized, and a certain agent which will fulfill the needed actor role. These relationships are specified by instances of fact types: *[organization artifact bundle] of [agent training]; [agent training] of [organizational engineering process]; [agent] of [agent training]*. A certain agent training will be part of exactly one OEP and consist in the training of exactly one agent with exactly one organization artifact bundle, thus, the

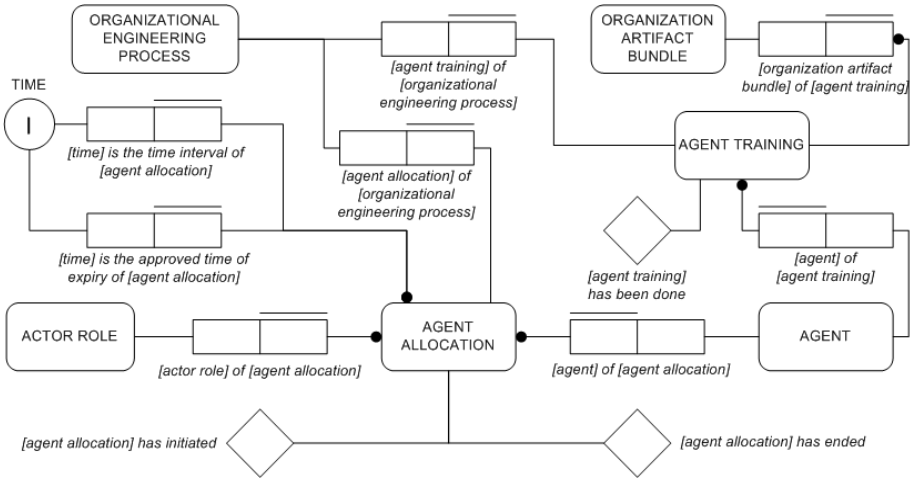

Fig. 5. G.O.D. Organization's SSD – part 3

specified unicity laws. Concerning the 10 organization artifacts previously defined as members of bundle OAB01, after training a certain agent (or agents) with the devised specifications the operationalization is realized with the allocation of such agent(s) to fulfill the relevant actor role(s). Thus, we also have object class AGENT ALLOCATION where instances of this class will be part of exactly one OEP and related with exactly one agent, one actor role, one time instant of expiry of such allocation and one time interval where such allocation is valid (it can be a composite time interval, like a normal work schedule with breaks for lunch and weekend break). The presented unicity laws are derived from the previous explanation and the expressed relationships are specified by instances of the following fact types: *[agent allocation] of [organizational engineering process]*; *[agent] of [agent allocation]*; *[actor role] of [agent allocation]*; *[time] is the time interval of [agent allocation]*; *[time] is the approved time of expiry of [agent allocation]*.

The specifications of a certain organization artifact bundle need to be devised, down to the implementation model, which is the most detailed specifications needed to realize operation [2]. To be operationalized, these specifications need to be implemented in real physical organization agents: human or IT agents (or a mixture of both). These agents are then allocated to actor roles so that these can be fulfilled. Implementing specifications in human agents means some kind of training like personal interactive training or giving a text manual or even the relevant organization artifacts themselves in some representation like diagrams, tables or action rules. Implementing specifications in IT agents means coding – e.g., creating program from scratch – parametrization – e.g., configuring COTS (Commercial Off-The-Shelf) or ERP (Enterprise Resource Planning) software – or a mixture of both. Although it is quite acceptable to say that one implements specifications in IT resources, it is rather inappropriate to say one implements specifications in human resources. We thus propose the adoption of the more neutral terms “training” – as a synonym of implementation – and “agent” – as a synonym of a human or IT resource constituting actor technology.

4 Conclusions

In the reality of an organization facing unexpected exceptions, microgenesis dynamics occur in the form of what we call organizational engineering processes where there is an intertwining play between three major categories of acts: unexpected exception handling acts, organizational artifact state change acts and organizational artifact operationalization acts. The trouble is that, like we saw in our problem definition, in most cases these acts and their respective facts (reflecting analysis and change of organizational reality) are not captured and are forgotten. For a certain OEP, instances of all fact types defined in the GO's SM can collectively be considered as the full description of the ad-hoc original set of actions and facts that constitute the execution of the OEP itself and its effect on the organizational self. One major contribution of our proposal of extension to DEMO is that our approach provides contextual information to organization artifacts whereas in DEMO's traditional "static" view one does not have such information. The state base of the GO will be able to provide, for a certain organization artifact, information like: the dysfunction that was happening and respective exception that led to its generation and operationalization; who did what and decided what in the respective OEP. As we have seen, such information may be quite valuable in future change contexts affecting such artifact.

We have presented our proposal for a G.O.D. Organization existing in every organization and responsible for the generation, operationalization and discontinuation of organization artifacts that constitute the organization self, i.e., its composition, structure and production. We gave more focus to the aspect of handling unexpected exceptions, showing how keeping structured historical information of dynamics of this handling can be quite useful to handle exceptions occurring in the present. We then showed how the G.O.D. Organization makes a bridge between the worlds of model and meta model, where the latter contains the set of generic organization artifact (meta level) types out of which a set of (model level) instances can be generated that constitute an organization. With our proposal of the G.O.D. Organization, DEMO no longer is limited to a "static" picture of an organization and we can now have a full trace of the state of the organization system. The current picture of the organization, or, in other words, its ontological model, simply consists in the conceptualization of the set of organization artifacts that are current, i.e., whose last event was "has been operationalized". We then briefly touched on the aspect of operationalization of organization artifacts, an issue that seems to also fall in the responsibility of the G.O.D. Organization. Operationalization implies training IT or human agents with the devised specifications and then allocate such agents to fulfill the necessary actor roles. The state base of the G.O.D. organization seems to be the right place to keep state information of who is responsible to fulfill what and who was responsible for generation and training of organization artifacts. For space reasons many aspects of the specification of the G.O.D. Organization were left out of this paper. For example, all specified result kinds will have their associated transaction kinds and actor roles as well as associated action rules. But the most important part was addressed here: the State Model.

As future research, we foresee that what we call an OEP is a particular case of a more generic Change Process, which can include other particular cases like Improvement Handling Process or Innovation Handling Process. The G.O.D. Organization

that we propose is surely incomplete and it can be extended to include the aspect of proactive change, whereas we have focused, on this paper, on reactive change. Another issue to be addressed in future research is practical tools and guidelines to implement the G.O.D. organization as, in our research, we have mostly tread the conceptual and proof of concept aspects.

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.

References

1. Applegate, L.M., McFarlan, F.W., McKenney, J.L.: Corporate information systems management: text and cases. McGraw-Hill, Irwin (1999)
2. Dietz, J.L.G.: Enterprise ontology: theory and methodology. Springer-Verlag New York, Inc., Secaucus (2006)
3. Christensen, W.D., Bickhard, M.H.: The process dynamics of normative function. The Monist 85, 3–29 (2002)
4. Holland, J.H.: Hidden order: how adaptation builds complexity. Basic Books, New York (1996)
5. Bickhard, M.H.: Error dynamics: the dynamic emergence of error avoidance and error vicariants. Journal of Experimental & Theoretical Artificial Intelligence 13, 199–209 (2001)
6. Brown, S.L., Eisenhardt, K.M.: Competing on the edge: strategy as structured chaos. Harvard Business School Press, Boston (1998)
7. Saastamoinen, H., White, G.M.: On handling exceptions. In: Proceedings of conference on Organizational computing systems, pp. 302–310. ACM, New York (1995)
8. Mourão, H.: Supporting effective unexpected exceptions handling in workflow management systems within organizational contexts. Science Faculty of Lisbon University (2007)
9. Tribolet, J.: Organizações, pessoas, processos e conhecimento: da reificação do ser humano como componente do conhecimento à "consciência de si" organizacional [organizations, people, processes and knowledge: from the reification of the human being as components of knowledge to the knowledge of organizational self]. Sistemas de informação organizacionais. Sílabo Editora. Lisbon, Portugal (2005)
10. Magalhães, R., Zacarias, M., Tribolet, J.: Making sense of enterprise architectures as tools of organizational self-awareness (osa). In: Proceedings of the Second Workshop on Trends in Enterprise Architecture Research (TEAR 2007), June 2007, vol. 6, pp. 61–70 (2007)
11. Zacarias, M., Magalhães, R., Caetano, A., Pinto, H.S., Tribolet, J.: Towards organizational self-awareness: an initial architecture and ontology. In: Handbook of ontologies for business interaction. Information Science Reference, pp. 101–121 (2007)
12. Magalhães, R., Silva, A.R.: Organizational design and engineering (ode) - ode white paper - version 1 (2009)
13. Casati, F., Pozzi, G.: Modeling exceptional behaviors in commercial workflow management systems. In: Proceedings of the Fourth CoopIS - International Conference on Cooperative Information Systems, pp. 127–138 (1999)

14. Axelrod, R., Cohen, M.D.: Harnessing complexity: organizational implications of a scientific frontier. Basic Books, New York (2001)
15. Dietz, J.L.G., Albani, A.: Basic notions regarding business processes and supporting information systems. Requirements Engineering 10, 175–183 (2005)
16. Op' t Land, M.: Applying architecture and ontology to the splitting and allying of enterprises. TU Delft (2008)
17. Dietz, J.L.G.: On the nature of business rules. Advances in Enterprise Engineering I, 1–15 (2008)
18. Bunge, M.A.: Treatise on basic philosophy, a world of systems, vol. 4. Reidel Publishing Company (1979)
19. Bon, J.V.: It service management, an introduction. itSMF-International (2002)
20. Dietz, J.L.G.: Demo meta model specification (forthcoming) (2009), `http://www.demo.nl`
21. Dietz, J.L.G.: A world ontology specification language. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM-WS 2005. LNCS, vol. 3762, pp. 688–699. Springer, Heidelberg (2005)