Towards Organizational Modules and Patterns based on Normalized Systems Theory

Peter De Bruyn, Herwig Mannaert and Jan Verelst
Normalized Systems Institute (NSI)
Department of Management Information Systems, University of Antwerp
{*peter.debruyn,herwig.mannaert,jan.verelst*}*@uantwerp.be*

***Abstract*—Normalized Systems Theory provides prescriptive guidance on how to design modular software structures so that they exhibit a high degree of evolvability and low degree of diagnostic complexity. The theory basically suggests to first break up the modular structure in a very fine-grained way based on "concerns" (in terms of change drivers or information units) and then aggregate these concerns in a structured way into patterns or "Elements". While the relevance of this theory for analyzing and designing organizational artefacts has been shown previously, only the first step of the reasoning has already been performed in the past, i.e., identifying a set of organizational concerns to be separated. In this paper, a first attempt to proceed to the second step (i.e., aggregating concerns into organizational "Elements") is proposed. We formulate some meta-requirements for such organizational Elements (i.e., having exhaustive interfaces, aggregating several basic constructs into one "Element" as well as including and identifying a relevant set of cross-cutting concerns). We also propose a tentative set of five organizational Elements: Party, Product or Service, Compensation, Work Unit and Asset or Resource. The relevance of these Elements is shown by briefly discussing some (theoretical) illustrations.**

***Keywords*-Normalized Systems; Organizational patterns; Interface definition.**

I. INTRODUCTION

Modularity —dividing a system into a set of interacting subsystems— has proven in the past to be a powerful concept in order to describe or build artefacts (such as software, organizations, etc.) so that they exhibit a set of advantageous characteristics, such as evolvability or lower diagnostic complexity. Increased evolvability, as it allows to adapt only a part (i.e., one module) of the artefact while leaving the other parts unchanged. Lower diagnostic complexity, as for instance erroneous outcomes can be traced to a particular subsystem (i.e., one module) instead of leaving the whole system to be analyzed for its cause. However, prescriptive guidance regarding how to design software systems or organizational artefacts as modular structures exhibiting these properties is rare. Normalized Systems Theory (NST) has recently proposed such prescriptive guidance to develop highly evolvable systems [1] with low complexity [2]. As a first step, the theory proposes a set of theorems which basically prescribe how to fragment a modular system (i.e., based on concerns such as change drivers or information units) for this purpose. In a second step, a set of patterns (called "Elements") are proposed to realistically apply the theorems in practice. Both steps have already been performed and documented at the software level (on which NST was originally applied). At the organizational level however, only the relevance of the theory ([3], [4]) as well as some initial guidelines to partition business processes [4] as organizational artefacts (i.e., step 1) have been suggested. Therefore, no set of patterns (or "Elements") at this organizational level are available.

In this paper, the authors propose a first attempt regarding the formulation of such potential Elements at the organizational level in order to make an initial contribution to this research gap. We will first briefly discuss some essential aspects of NST, showing how the Elements at the software level have been conceived (Section II). From this discussion, we derive a set of meta-requirement which should be fulfilled when formulating Elements at the organizational level in Section III and propose a potential initial set of five organizational Elements in Section IV. We then provide some illustrations (Section V) and offer our final conclusions (Section VI).

II. NORMALIZED SYSTEMS THEORY

In a quest for making systems more evolvable and less complex, NST has applied the concepts of stability from systems theory ([1], [5]) and entropy from thermodynamics [2] to modular systems and derived a set of formally proven *design theorems* prescribing how to identify and delineate individual modules. For instance, from the evolvability point of view, the Separation of Concerns theorem prescribes that every change driver (i.e., part of system which can independently change) should be isolated into its own modular construct (e.g., class in a object-oriented software environment). NST further shows that each of its principles should be systematically applied throughout the whole modular structure in order to truly realize its benefits. As this results in highly fine-grained modular systems, it is a very challenging effort to manually design a system in reality without any theorem violations.

Consequently, as a second step in its reasoning, NST proposes to employ a set of patterns (called "*Elements*"),

 ISBN: 978-1-61208-319-3

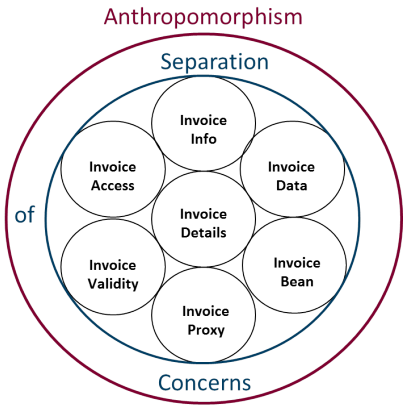

Figure 1. A NST Data Element (based on [5]).

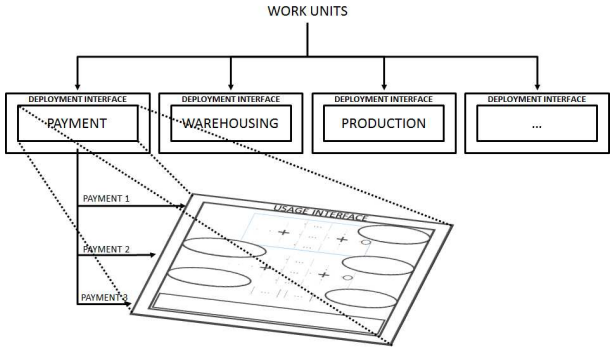

Figure 2. A schematic representation of an Organizational Work Unit deployment.

which are a structured aggregation of modular structures, conforming with the NST theorems and which can be more readily used to build real-life applications. At the software level, five elements were proposed: Data Elements, Action Elements, Workflow Elements, Connector Elements, and Trigger Elements. Experience has demonstrated that this set of elements indeed enables the creation of real-life (business or administration oriented) software applications [6]. Figure 1 represents the internal structure of one of these Elements, i.e., a Data Element for storing and exchanging information regarding an Invoice. Typically, such Element consists out of one core construct (here: software class), which performs or stores the main functionality of the Element (in this case: a set of data attributes, for instance the class "InvoiceDetails" for an Invoice Data Element). On top of this core construct, several other constructs are included within the Element taking care of the cross-cutting concerns. In case of this software Data Element, such classes are added in order to take care of persistency (e.g., "InvoiceData"), transactional integrity (e.g., "InvoiceBean"), etc. In essence, each of these classes incorporating such cross-cutting concern represents a "mini-bus" acting as a proxy or facade to the (external) technology used for this cross-cutting concern (e.g., JPA, EJB). Due to the NST theorems, the parts implementing a cross-cutting concern should be separated within their own construct inside the Element. Such design also ensures version and implementation transparency regarding functional changes (e.g., adding a new data attribute or updating the technology of one of the cross-cutting concerns) as the impact of these changes remains isolated within one instantiated Element.

While originally applied at the level of software systems, it has been shown that NST reasoning can be applied to organizational artifacts, such as business processes [4] or enterprise architectures [3] as well. More specifically, a set of guidelines has been proposed in order to help identifying typical concerns at this organizational level, which should therefore be separated [4]. This again results in a very fine-grained modular structure, difficult to arrive at each time "from scratch". Consequently, a set of Organizational Elements might be beneficial at this level as well. However, such Elements have not yet been proposed.

III. The Meta-Requirements of Organizational NST Elements

Elements at the organizational level would imply the creation of organizational modules, which exhibit the same characteristics as the Elements at the NS software level. Therefore, we discuss three necessary characteristics for organizational "chunks" to become modules and even Elements eventually exhibiting high evolvability and diagnostability.

A. Defining exhaustive interfaces

Earlier, we defined modularization as "*the process of meticulously identifying the dependencies of a subsystem, transforming an ambiguously defined 'chunk' of a system into a clearly defined module (of which the borders, dependencies, etc. are precisely, ex-ante known)*" [7]. In doing so, we differentiated our definition from that of Baldwin and Clark [8] in which a module should already exhibit high intramodular cohesion and intermodular coupling. While we acknowledge that such properties are clearly highly valuable characteristics for modular structures, we stress the importance of exhaustively defining a subsystem's interface (i.e., its interaction with its environment) before engaging in more sophisticated optimization efforts such as maximum cohesion and minimum coupling. In order to arrive at such complete interfaces for modules at an organizational level, we highlight the following aspects which should — at minimum— be taken into account when one's aim is to arrive at blackbox modules having a fully defined interface.

First, a distinction could be made between those parts of an interface which are related to the "set-up" of the module (the so-called "*deployment interface*") versus those parts related to the calling or invocation of the module (the

so-called "*usage interface*"). The deployment interface can be understood as (mostly) a "one-time" interaction to get the module initially "deployed" and operational in its working environment (e.g., the necessary infrastructure), whereas the usage interface is related to the repetitive interaction each time a request (input) to produce a certain result (outcome) is directed towards the module. The difference between both types of interfaces is visually depicted in Figure 2. Second, both the "functional" or "front-end" interface as well as the "non-functional" or "back-end" interface should be recognized. The *"functional" interface* typically relates to the input arguments given to the module to provide a certain product or service (e.g., the amount of money to be transfered by a payment module). The *"non-functional" interface* concerns these interactions (or parameters) which are not necessarily viewed as directly linked to the product related input or output provided by a module, but are nevertheless necessary to obtain its correct execution (e.g., the presence of a well-functioning Internet connection). Often, this functional interface aligns with the usage interface whereas the non-functional interface mostly corresponds to the deployment interface. As we see that, in practice, the "deployment" and "non-functional" interfaces are often not explicitated, while necessary to obtain true black box modules, we suggested in the past some interface dimensions which should be taken into account in order to obtain exhaustively defined interfaces for organizational modules such as:

- *Supporting technologies* (e.g., SWIFT, the Internet);
- *Knowledge, skills and competences* needed by people carrying out tasks in the organization;
- *Money and (financial) resources* required to operate the module;
- *Human resources, personnel and time* defining how many (concurrent) people are required (and for how long) in order for carry out certain activities;
- *Infrastructure and real estate* required to perform the activities (e.g., offices, machines);
- *Other modules or information* needed for a module to operate correctly.

B. Aggregating modular building blocks into organizational Elements including cross-cutting concerns

Experience with NST at the software level has shown that obtaining evolvable and diagnosable modular structures, requires a very thorough separation of concerns resulting in a very fine-grained modular structure. However, obtaining such systems in practice only seems realistic when the designer has a set of "Elements" at his disposal, each being a structured aggregation (i.e., pattern) of modular building blocks. At the software level, the modular building blocks available to compose Elements are typically classes and objects in object-oriented environments or structures, functions, procedures in structured programming languages. In organizations, such basic modular building blocks might constitute people, the actions they perform (and their order), as well as the materials they need and/or produce. The formulation of the NST Elements based on these building blocks is likely to be an inductive process (i.e., the Elements eventually have to comply to the NST theorems, but are not directly derivable from them) and the union of all NST Elements should provide all basic functionality to build state-of-the-art modular structures within its application domain. At the software level, Data, Action, Flow, Connector and Trigger Elements were suggested which allow to store data, perform actions with them, define an order of sequence between these actions, connect with external systems in a stateful way and trigger the necessary Data and Flow Elements on a time-based manner if needed, respectively. Therefore, structured aggregations of modular building blocks might equally result in Elements at the organizational level, for instance the five Elements we will suggest as potential candidates in Section IV.

However, we showed in Section II how NST incorporates a set of classes in these Elements as proxies or facades to a number of cross-cutting concerns. The concept of cross-cutting concerns was already well-known in the software engineering field when adopted by NST. For instance, in the "aspect-oriented" programming paradigm, attention to the principle of separation of cross-cutting concerns was already given. Nevertheless, in many other application domains in which modular designs are present, such cross-cutting concerns seem to be useful as well. Consider — as a simple example— the modular design of a house. Each of the rooms (e.g., living, kitchen, bedroom) could be considered as a separate module, each having their own functionality. However, while there might be some central and separated water, heating and electricity supply and management systems, these facilities typically have to be available throughout the whole house. Each room wants to have the possibility of tapping water, using the heating system and employing electrical devices. As a consequence, electric cords with sockets, water pipes with taps and heating pipes with radiators are incorporated in each module (here: house room), connected with the central facility provision units. Therefore, the water, heating and electricity facilities in a typical house can be considered as quite adequate illustrations of cross-cutting concerns in the construction area. As organizations have been argued to be modular structures, several modular structures seem to have "cross-cutting concerns" and NST states that these cross-cutting concerns should be incorporated in a structured way into Elements in order to make the structures evolvable and lowly complex, it seems interesting to incorporate such cross-cutting concerns in organizational Elements as well. The question then obviously becomes: can we identify cross-cutting concerns at the organizational level, and of what kind are they? Once we have some insight into this aspect, we

 ISBN: 978-1-61208-319-3

can start to formulate Elements at the organizational level.

C. Identifying cross-cutting concerns

NST reasoning suggests that identifying cross-cutting concerns at the organizational level would allow us to obtain highly evolvable and lowly complex artefacts in this domain. First, the concept of "cross-cutting concerns" in organizations seems an appealing thought intuitively. Indeed, one can easily imagine the necessity for most parts or modules in an organization to communicate with items external to the module, or report relevant items in the bookkeeping system. Second, certain business modeling languages seem to acknowledge the importance of such cross-cutting concerns in an implicit way: to keep the modeling and mental complexity manageable, they make abstraction from certain aspects during their modeling efforts while concentrating on mostly one relevant aspect regarding each functional domain of an organization, thereby reflecting their "view" on the functioning of organizations. For instance, regarding the bookkeeping aspects of an organization, the "Resources, events, agents" (REA) approach provides a method to design accounting information systems [9]. As a consequence, an organization is modeled as a set of economic events (performed by economic agents) resulting in stock flows in terms of economic resources. As this is the primary focus of the modeling method, other organizational aspects are mainly put out of scope. Another example, regarding the communication aspects of an organization, involves the Enterprise Ontology (EO) theory and its accompanying DEMO method, claiming that the essence of an organization resides within a generally recurring communication pattern regarding ontological actions (i.e., involving human decisions or the creation of products) [10]. As a consequence, an organization is primarily modeled as a set of actors engaging in the request-promise-execution-statement-acceptance (standard) pattern of ontological (communication) acts and facts, abstracting away from other organizational aspects (e.g., the implementation of the execution step).

Without claiming to have the optimal or even an exhaustive set of cross-cutting concerns, we suggest the following concerns as being some suitable candidates of cross-cutting concerns for the "Elements" at the organizational level (all of them will be incorporated in one of our proposed Elements, i.e., the Work Unit Element):

- **Registration or logging**: As tasks and their encompassing flows are executed, relevant information should be logged and registered. Multiple different kinds or "dimensions" of relevant information could be thought of in this respect, e.g., throughput time, resource consumption, quality metrics, etc. Therefore, in order to reduce complexity during and after execution time, this information should ideally be logged at the fine-grained level of information units, i.e., each part within a business process design of which independently traceable information according to these dimensions is assumed to be needed later on [11]. This information should be persisted in one way or the other, which is the responsibility of the constructs making up this cross-cutting concern. Such information may then be used (and possibly aggregated) later on for diagnostic, KPI-reporting, business intelligence and other purposes.
- **Transactional integrity (including cancellations)**: On top of registering certain properties regarding information units during the execution of task(s) (flows), some modules should be able to handle requests from outside, as well as internally triggered actions based on stateful transactions (i.e., interacting life cycle information objects going through their respective life cycles). Amongst others, this means that a flow can only be in one state at a time, no conflicting or contradicting states should be able to occur and cancellation requests are handled in an appropriate way (e.g., no blind interruption of the regular or the "happy" path). These predetermined transaction structures should be enforced one way or the other, which is the responsibility of the constructs making up this cross-cutting concern.
- **External communication**: As the organizational modules are expected not to work in isolation, they should be able to communicate via incoming and outgoing information streams with other modules. However, as communication issues (such as message format, fault handling, background technology) clearly do not belong to the "core" responsibility of each genuine (i.e., non-communication dedicated) module, (the connection to) these issues should be handled in different construct instances than those which do handle this core responsibility. Consequently, to enable the interaction of a module with external modules, communication should be incorporated as a cross-cutting concern within organizational elements.
- **Authorization policy**: Typically, flows and their constituting tasks are only allowed to be executed by certain actor(s) (roles) due to safety, legal and company defined regulations. In addition, in certain cases, the actual identity of these actor roles should be verified. As the common role definitions and identity verifications should be accessed and used throughout several organizational modules, these concerns seem to depict a genuine cross-cutting concern as well, handling these functionalities.
- **Bookkeeping adapter**: Many executed tasks and flows result in one way or the other to changes in the bookkeeping ledgers or financial reporting systems of an organization. However, bookkeeping standards as well as the way in which certain goods and assets are valued, are clearly a different kind of concern as the "core" activities of, for instance, a procurement or assembly module. Therefore, some constructs should

 ISBN: 978-1-61208-319-3

be able to extract the information needed for several bookkeeping standards (based on the logged or registered information) as a cross-cutting concern within many organizational modules, and provide it to central bookkeeping and financial reporting modules [12].

IV. Towards Normalized Systems Elements at the Organizational Level

Looking for the Elements which might be needed in order to fully describe the functioning of organizations, we adopt the following perspective. As depicted in Figure 3(a), the behavior of organizations can basically be described as a number of interacting Organizational Work Units, executing (flows of) tasks by employing a set of internal Assets or Resources. Eventually, these Work Units deliver Products and Services to Parties external to the organization (such as customers). The delivery or procurement of certain Products or Services is generally associated with an act in return: a Compensation. To the extent that this set of concepts allows us to model a domain (in this case: organizations) and explain some reasoning about it, it could be regarded as a kind of new, modest ontology regarding the functioning of organizations in terms of Elements. However, before being able to make this claim, we should more specifically define each of these concepts and articulate the relationships between them.

- **Party**: A Party is a (natural) person, company or legal entity which is doing business. Whereas we identify an Entity as anything from a company to an organizational department, we consider a Party as an identifiable "actor" which has the authorization to act on behalf of an entity. The core of the Party Element is the listing of its identification and idiosyncratic properties. Additionally, a set of cross-cutting concerns is added, which are typically needed for each Element of this type (e.g., external communication). For instance, a company procuring Bike Parts, assembling them to Bikes and delivering them to customers, could be considered as a Party (as well as its suppliers, customers and employees under contract).
- **Product or Service**: A Product (typically tangible) or Service (typically non-tangible) is something an organization (which is a Party) is capable of providing to its customers (which are also Parties). The core of the Product or Service Element comprises the definition (its characteristics) and nature of the Product or Service (i.e., production details, marketing details, etc.). Moreover, it contains some links to Organizational Work Units (cfr. infra) performing certain (sets) of tasks necessary to actually provide the Product or Service according to its characteristics defined in the "core". For instance, in the case of a bike producing company, a Product could constitute a Bike so that Customer ABC could request Company XYZ to produce a particular Bike instance with chassis number 123. While delivering this Bike instance, the Bike company will most likely rely upon several of its Work Units (e.g., Procure, Assembly).
- **Compensation**: A Compensation is an act in return from one Party to another Party, generally because the latter Party has received or will be receiving a certain Product or Service from the first one. As a consequence, a Compensation is typically linked with a Product or Service provided by one Party. It is important to note that a Compensation might take on several forms (e.g., an invoice, followed by a payment, etc.), not necessarily being of a financial kind. Additionally, a Compensation might have instantiation frequencies which are not necessarily completely aligned with the instantiation frequency of the associated Product or Service. Therefore, the core of the Compensation Elements consists of the definition and properties (i.e., the "terms") of the Compensation, supplemented with a set of cross-cutting concerns to complete this Compensation according to its specified terms. For instance, a Bike producing company could issue a Compensation instantiation with a customer as a result of delivering a Bike to that Customer earlier. Alternatively, the company might have to fulfill a monthly recurring payment procedure with its bank as part of its loan for its real estate ownership. As a consequence, also a Compensation is typically associated with one or more Organizational Work Units (cfr. infra).
- **Asset or Resource**: Assets and Resources are defined as both human (e.g., employees) and non-human parts (e.g., product parts, consumable raw materials, buildings), which are used for carrying out tasks and/or flows. We consider Assets or Resources as internal to an organization. However, these may be acquired or hired via contracting or purchases between Parties earlier in time. Therefore, the core of the Asset or Resource Element consists of the Asset or Resource identification and its characteristics, as well as a set of cross-cutting concerns to generally employ the Asset or Resource (e.g., communicate with it). For instance, the Assets or Resources for a Bike producing company might constitute of a set of employees (sales people, assembly people, etc.) and infrastructure (building, manufacturing equipment). In order for the company to employ these Assets or Resources, the company might have bought the building and have hired the employees resulting in a Compensation instantiation.
- **Organizational Work Unit**: An Organizational Work unit is the whole of a set of tasks, task flows and accompanying assets/resources needed to execute these tasks. A *Task* is a small part of work (performed by resources/assets such as human actors or machinery, and potentially consuming other resources/assets), iden-

 ISBN: 978-1-61208-319-3

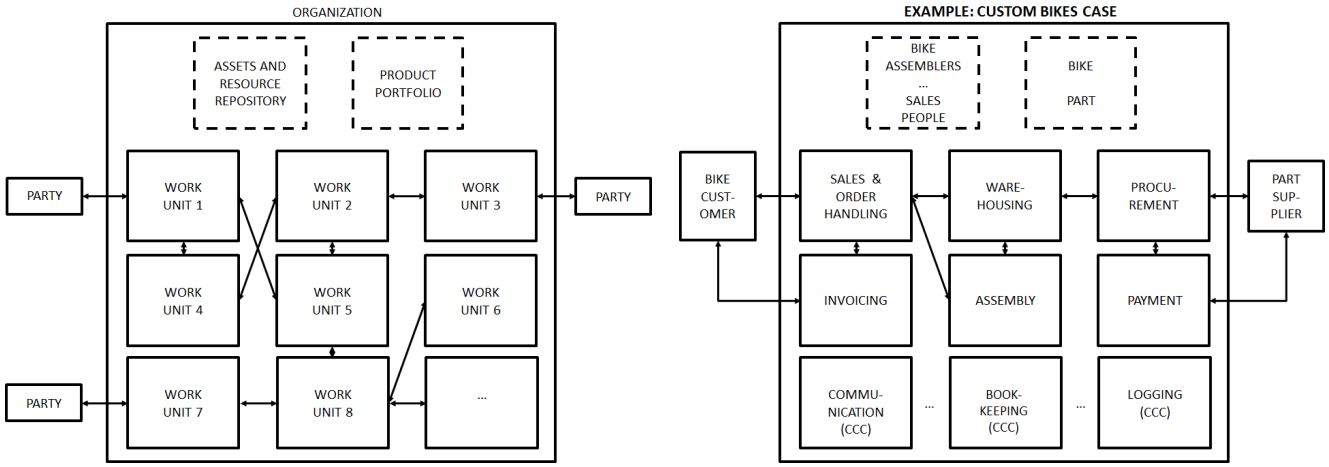

Figure 3. An organization modeled as the interaction between a set of deployed Organizational Work Units, using Assets and Resources, producing Products and Services and generating Compensations. Eventually, the Products and Services, as well as Compensations are exchanged with Parties external to the organization.

tified as being a separate change driver or information unit. A *Task Flow* is defined as a set of tasks in a particular order (including sequences, iterations and selections) all operating on one lifecycle information object. The internal design of the flows and tasks within the Organizational Work Unit should adhere to the guidelines for designing evolvable [4] and lowly complex business processes ([11], [13]), as documented before. Ideally, this core of the Organizational Work Unit (i.e., tasks and flows, complemented with a set of Assets or Resources) should also constitute a logical whole which is externally loosely coupled. However, in contrast with an exhaustively defined interface, this is not a prerequisite as a full interface enables the analysis and optimization of the delineation of these Work Units. Some relevant cross-cutting concerns for an organizational Work Unit were already discussed in Section III-C.

To sum up, we propose a Party, Product/Service, Compensation, Assets or Resources and Organizational Work Unit as a tentative set of Elements within the application domain of organizations. The Work Units use Task (flows) and Assets/Resources to perform their function. A potential internal structure (i.e., content) regarding the Products or Services, Compensation, Assets or Resources and Parties is shown in the various panels of Figure 4. Each of the Elements contain some core and basic information regarding that Element type (such as identification, description, etc.) as well as a set of cross-cutting concerns (visualized by the ovals) depicting some viable cross-cutting concerns for each of them. It is important that the proposed sets of cross-cutting concerns should not be considered as exhaustive or restrictive. Rather, our goal is to illustrate reasoning behind the aggregation mechanism for obtaining Elements, while incorporating cross-cutting concerns at the organizational level. As it is clear that the major part of organizational results is produced by the Organizational Work Units, the internal working of external organizations or people (i.e., Parties) is not to be manipulated, and Products/Services as well as Compensations largely depict the relation between multiple Parties, let us take a closer look at the internal structure (i.e., design of once it is deployed) of the Organizational Work Unit as depicted in Figure 5. As can be seen from the figure, we conceive an Organizational Work Unit as a set of individual tasks, a set of flows consisting of tasks, consuming assets and resources (e.g., raw materials, people performing the actions), incorporating the above enumerated set of cross-cutting concerns and surrounded by the appropriate (exhaustive) usage interface.

There are basically two main reasons why we chose to start from our own set of ontological concepts, instead of starting from one of the several already existing ones in extant literature. First, as we already argued in Section III-C, many existing business modeling methods in literature seem to mainly focus on one particular aspect (e.g., one of our suggested potential cross-cutting concerns for organizational modules). Second, we did not want to start our discourse on NST Elements at the organizational level by criticizing or claiming to ameliorate one particular existing organizational ontology. Nevertheless, we are aware that other ontological frameworks might have similar or related concepts (e.g., many frameworks acknowledge the existence of things like “actors” or “parties”), but the set of our five proposed Elements and their accompanying interpretation is —to the best of our knowledge— new. Also, we think that our proposed ontology largely aligns with the “common denominator” or a

 ISBN: 978-1-61208-319-3

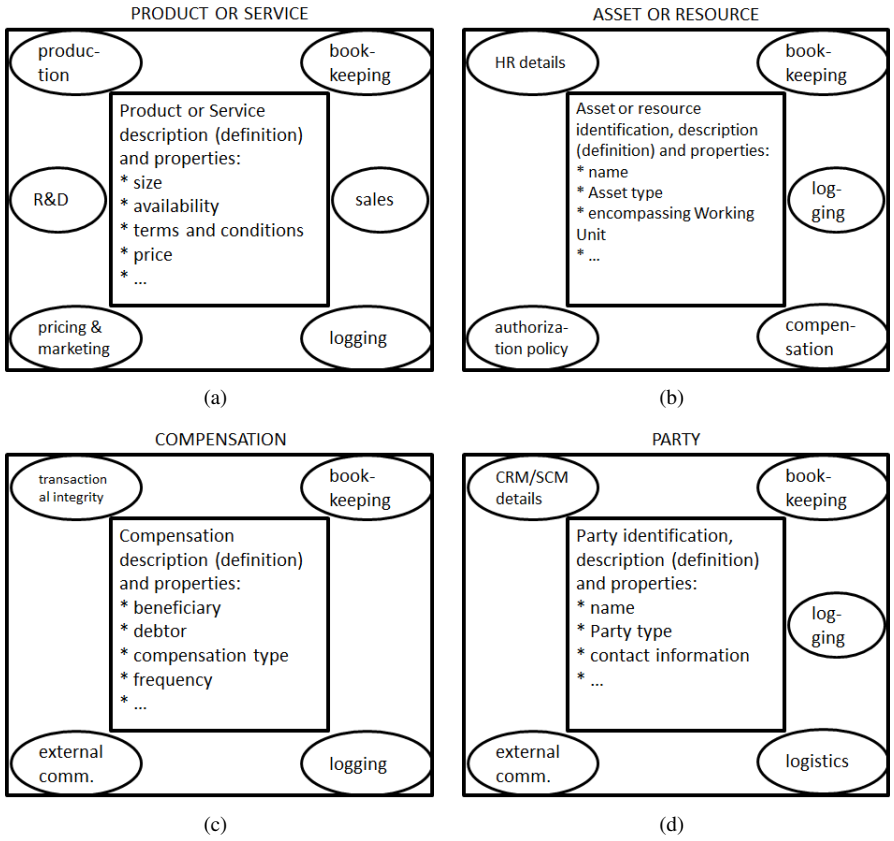

Figure 4. Organizational Product/Service, Asset/Resource, Compensation and Party Elements.

lot of "heuristic feeling" when conceptualizing about organization (just as our NST ontology at the software level did). An organization basically consists of Assets or Resources carrying out task and their flows (eventually aggregated into modules), in order to deliver Products or Services to Parties in return for some Compensation. However, we encourage other researchers to compare our NST reasoning with their own business modeling efforts and look for opportunities of cross-fertilization between both. For instance, we hypothesize that it should be perfectly possible to combine (certain parts of) our NST reasoning on organizational Elements with some of the already existing ontological frameworks (such as EO, REA, etc.).

Consequently, equally as for the "Elements" previously proposed at the software level, we are fully aware that our initial ontology (both the set of elements as well as their internal structure) is only one possibility of modeling and modularizing a company, and that other ontologies or Element structures are possible, and might potentially even be more optimal. Rather, the main purpose of our proposed organizational Elements is to provide a constructive proof-of-concept regarding the feasibility of modeling and designing organizations realistically in a way adhering to the NST theorems. However, the authors want to emphasize that our organizational elements seem to allow modularization in the three dimension as mentioned by Campagnolo and Camuffo (i.e., product design, production system and organizational design) [14]. This indicates that realistically deployable organizational modules seem to be at the intersection of business processes and organizational departments, thereby combining several viable modularization dimensions within organizations (e.g., according to functional domains, mainly process-oriented, etc.). Consequently, modularization as conceived by NST Elements does not necessarily imply extreme specialization due to its fine-grained separation: at the level of an Element, each module has its own "core" responsibility, supplemented with a link to a variety of cross-cutting concerns.

V. ILLUSTRATIONS

In this section, we will illustrate our reasoning from above in two ways. First, from a more high-level approach, we will show how nearly all aspects of an exemplary organization (labeled as "the Custom Bikes case" by Van Nuffel [4]) can be described or categorized while employing our proposed ontology. Second, from a more bottom-up approach as suggested by NST, we will elaborate on the possible internal design of one possible Organizational Work Unit, a Payment

 ISBN: 978-1-61208-319-3

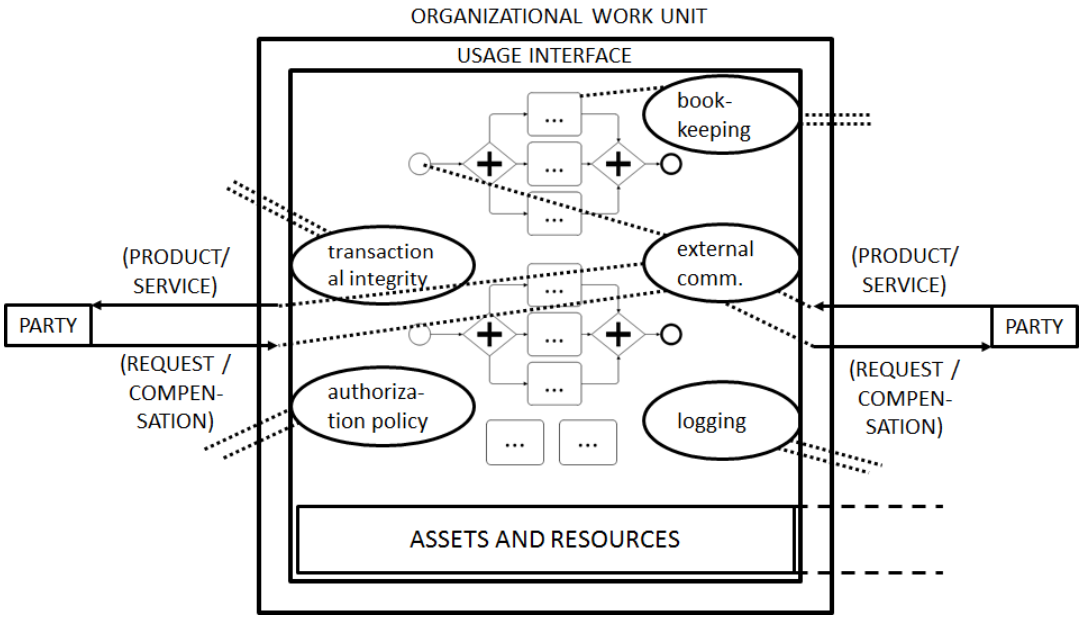

Figure 5. A schematic representation of an Organizational Work Unit, an Element at the organizational level according to NST reasoning.

Element.

A. Applying the Organizational Constructs and Elements

The "Custom Bikes case" [4] concerns a small company producing customized bicycles. Basically, for each order which is received, an order handling process starts (e.g., checking the customer details, evaluating and categorizing the order, manager approval, order scheduling). If needed, a process can be executed to order some missing parts, after which the assembly of the bike is done, shipped and invoiced. The result of applying our organizational constructs to this organization is shown in Table I and Figure 3(b). We can notice that nearly all aspects of the case can be categorized within our set of ontology constructs. In the work of Van Nuffel [4], most efforts have been invested in the identification and modeling of the tasks and flows. While one can argue that some initial attention has also been given to Products or Services, Compensations and Assets or Resources, the modeling of Parties and (often interface-related) Assets or Resources seems to be abstracted away in the work. Additionally, no Organizational Work Units according to our definition can be discerned. In essence, this means that a thorough fine-grained separation of concerns was performed regarding task (flows), but that no patterns or organizational "Elements" were explicitly identified. As we explained earlier, this type of fine-grained modularization is a first important step in the normalization of modular structures. However, the systematic definition of an exhaustive interface, incorporation of cross-cutting concerns, as well as a structured aggregation of the flows of activities (together with its Assets or Resources) should, in our opinion, be performed in a next stage in order to arrive at more "realistic" organizational modules and "Elements".

Table I
APPLYING THE ORGANIZATIONAL CONSTRUCTS AND ELEMENTS

Construct type candidates	Manifestations in Case
Product or Service	Bike Part
Compensation	Invoice Payment
Party	Customer Employee Part Supplier Custom Bikes Company
Asset or Resource	Sales people Bike Assemblers Warehouse employer ... Manufacturing plant building Manufacturing equipment Electricity supply ...
Tasks and their flows	Order Handling flow Customer Entry flow Purchase Order flow ...
Organizational Work Units	To be identified

B. A Design Case: Towards a Payment Element

Let us consider the inner design of a possible Payment Element, responsible for carrying out the payments in an organization, to demonstrate the viability of an organizational Work Unit in reality. If we start by drafting a **deployment interface** of such Element, we see that some Assets/Resources are required to operate the module. First,

we need a *human person* to operate some of the task(flows) within the Element, for instance expressed as a percentage of full-time equivalents. Some *IT infrastructure* (e.g., servers, PC) might be needed, as well as a connection to *external services* completing the payment (e.g., Internet, Swift, Isabel). Finally, we probably need some *infrastructure* as well (buildings, desk, etc.). Ideally, this deployment interface exhibits version transparency, meaning that if the Payment Element or the Elements using the Payment Element are changed (i.e., they go to a new version) the deployment interface remains unchanged (i.e., no additional Assets/Resources need to be deployed in order to ensure the proper functioning of the Element).

Realizing the deployment interface is similar to a "constructor" in a software environment, in the sense that these Assets/Resources become available for the deployed and implemented module. As we consider Assets/Resources as Elements as well, each of their instantiations is again encapsulated within their own set of cross-cutting concerns. For example, the employment contract with the employee is handled by the cross-cutting concern making the link to a Compensation. The needed bookings in the bookkeeping records for employing an Asset/Resource are handled by the bookkeeping adapter cross-cutting concern (e.g., the yearly depreciation of a building) and any changes in their "state" (e.g., extra courses followed by an employee) are followed up and tracked by the logging cross-cutting concern. This also implies that a particular Work Unit Element could make a request to such Asset/Resource Element instance to receive the information about its yearly costs (e.g., in terms of depreciation or wage) for (for instance) their own cost accounting purposes.

In terms of the **usage interface**, each individual payment to be carried out is most likely associated with a *Compensation* stipulating a certain amount to be paid by the company at a certain point in time in a certain currency to a certain account. Also this interface should ideally be version transparent so that a new version of a Compensation can still be processed by the existing Payment Element and that new implementations of the Payment Element can still process the existing Compensations. The actual execution of this payment can be typically seen as a set of (orchestrated) *tasks*, being part of a *(sub)flow*. While some of these tasks will be performing a genuine part of the payment execution logic (e.g., perform a data check, verify amount of current bank account, initiating the electronic transaction, etc.), some of them will use the proxy or facade constructs within the Element triggering a cross-cutting concern. For instance, the *authorization* cross-cutting concern can be used to verify whether only entitled persons carry out payments (e.g., only managers are allowed to trigger payments over 1000 euro). The *logging* cross-cutting concern would log (for instance in a database) for every flow instance and task instance, when it is performed, by whom, how long it took, which resources were consumed etc. The *transactional integrity* cross-cutting concern might use some rules to make sure that cancellation requests for a Payment are consistently rejected or might be performed when certain conditions are met, but would in any case not allow the uncontrolled interference of any cancellation request with the stateful and ongoing transactions within the Element. Finally, the *bookkeeping* cross-cutting concern of the Payment Element could make sure that the outflow of monetary resources is registered in the central bookkeeping and balance sheet. Informed about the completion of the Payment, the bookkeeping cross-cutting concern of the associated Compensation can make sure that this debt is removed from the balance sheet.

Internally, the modular design of such Element and its flow(s) is compliant with the NST guidelines for attaining stability and avoiding entropy. It should be clear that such element is not merely restricted to contain only one core flow or process. For instance, in case we should decide that the considered payment Element would not only contain a process to pay debts but also to check whether a claim from accounts receivable has been fulfilled or not. In that case, the "core functionality" of the Element would contain at least two processes.

Consequently, from a dynamic perspective, this Work Unit might cope with several **changes** in a stable way. First, the *resources* executing the task(flow)s might change (e.g., a human operator which gets replaced by a machine or software application), without affecting other Work Units as long as the usage interface remains unchanged. Second, the way a *cross-cutting concern* is implemented or the actor performing it, can easily be modified. For instance, if another bookkeeping standard should be adopted or a specified Party is appointed to take care of this cross-cutting concern (e.g., an external bookkeeper), the changes are isolated within this cross-cutting concern (and therefore not spreaded out among the core functionality flows of other Elements, which are just making the link via the proxies in which they are encapsulated). Third, specialization and optimization of this Work Unit could lead to the *re-use* of this Element within multiple organizational departments or even on cross-organizational scale. Indeed, one can imagine that such Payment module might not only cope with internal Compensations to be settled (i.e., coming from one's own company), but in fact any Compensation (as this could be the only prerequisite listed in the usage interface) including those from other organizations as well. Therefore, "performing payments" could become a re-usable Element or even a stand-alone business in the long term if wanted. Once again, this confirms our view on organizations as being essentially a (structured) aggregation of Work Units, coupled via exhaustive interfaces which are ideally loosely coupled.

Also, as the design requires very fine-grained information tracking, the degree of entropy can be controlled. For instance, additional cross-cutting concerns could be

 ISBN: 978-1-61208-319-3

imagined to be added to the Elements, later on. Consider a company wanting to perform *cost accounting*. In such case, an additional cross-cutting concern "cost accounting" could be incorporated, performing cost calculations based on the logged information (e.g., "500 invoices have been processed in one month, needing 1 FTE") and bookkeeping information (e.g., "this 1 FTE has cost of 3000 euro a month").

VI. Conclusion

This paper addressed for the first time explicitly the research gap regarding the NST Elements at the organizational level, similar to the Elements which exist at the software level. While we do not claim to have solved the research gap completely, the main contributions of this work should be situated in advancing the application of NST reasoning at the organizational level. More specifically, we discussed the general rationale and mechanism of such Elements, listed a set of three necessary meta-requirements for Elements at the organizational level (i.e., exhaustive interfaces, the aggregation of fine-grained modular building blocks, and the identification of cross-cutting concerns which should be included), and proposed an initial set of five Organizational Elements in this respect. In further (extended) work, we aim to discuss some more in-depth case studies based on our reasoning and look for potential issues during their realization. For example, while we are convinced that our reasoning regarding the formulation of exhaustive interfaces is theoretically solid, several issues might arise when trying to actually describe it in reality, determining the overall potential for realistic application. Additionally, it would be interesting to complement our work with domain related knowledge and expertise so that some of the Elements might become best-practice organizational artefacts which might be adopted by several organizations in the long term.

Acknowledgments

P.D.B. is supported by a Research Grant of the Agency for Innovation by Science and Technology in Flanders (IWT). Additionally, this paper is embedded within an IBM Faculty Award for "Applying Normalized Systems Theory at the organizational level".

References

[1] H. Mannaert, J. Verelst, and K. Ven, "The transformation of requirements into software primitives : studying evolvability based on systems theoretic stability," *Science of computer programming*, vol. 76, no. 12, pp. 1210–1222, 2011.

[2] H. Mannaert, P. De Bruyn, and J. Verelst, "Exploring entropy in software systems : towards a precise definition and design rules," in *Proceedings of the Seventh International Conference on Systems (ICONS)*, 2012, pp. 93–99.

[3] P. Huysmans, "On the feasibility of normalized enterprises : applying normalized systems theory to the high-level design of entreprises," Ph.D. dissertation, University of Antwerp, 2011.

[4] D. Van Nuffel, "Towards designing modular and evolvable business processes," Ph.D. dissertation, University of Antwerp, 2011.

[5] H. Mannaert, J. Verelst, and K. Ven, "Towards evolvable software architectures based on systems theoretic stability," *Software practice and experience*, vol. 42, no. 1, pp. 89–116, 2012.

[6] G. Oorts, P. Huysmans, P. De Bruyn, H. Mannaert, J. Verelst, and A. Oost, "Building evolvable software using normalized systems theory," in *Proceedings of the 47th Hawaii International Conference on System Sciences (HICSS)*, 2014, in press.

[7] P. De Bruyn and H. Mannaert, "On the generalization of normalized systems concepts to the analysis and design of modules in systems and enterprise engineering," *International journal on advances in systems and measurements*, vol. 5, no. 3&4, pp. 216–232, 2012.

[8] C. Y. Baldwin and K. B. Clark, *Design Rules: The Power of Modularity*. Cambridge, MA, USA: MIT Press, 2000.

[9] W. E. McCarthy, "The rea accounting model: A generalized framework for accounting systems in a shared data environment," *Accounting Review*, vol. 57, pp. 554–578, 1982.

[10] J. Dietz, *Enterprise Ontology: theory and methodology*. Springer, 2006.

[11] P. De Bruyn, P. Huysmans, H. Mannaert, and J. Verelst, "Understanding entropy generation during the execution of business process instantiations: an illustration from cost accounting," in *Proceedings of the Third Enterprise Engineering Working Conference (EEWC 2013)*, ser. Lecture Notes in Business Information Processing. Springer, 2013, vol. 146, pp. 103–117.

[12] P. Huysmans and P. De Bruyn, "Activity-based costing as a design science artifact," in *Proceedings of the 47th Hawaii International Conference on System Sciences (HICSS)*, 2014, in press.

[13] P. De Bruyn, D. Van Nuffel, P. Huysmans, and H. Mannaert, "Confirming design guidelines for evolvable business processes based on the concept of entropy," in *Proceedings of the Eighth International Conference on Software Engineering Advances (ICSEA)*, 2013, pp. 420–425.

[14] D. Campagnolo and A. Camuffo, "The concept of modularity within the management studies: a literature review," *International Journal of Management Reviews*, vol. 12, no. 3, pp. 259 – 283, 2009.

 ISBN: 978-1-61208-319-3