Gil Regev, Ilia Bider, Alain Wegmann

Defining Business Process Flexibility with the Help of Invariants

Software Process: Improvement and Practice (SPIP), V12(1), 2007, pp. 65-79

Defining Business Process Flexibility with the Help of Invariants

Gil Regev[1], Ilia Bider[2*], Alain Wegmann[1]

[1]Ecole Polytechnique Fédérale de Lausanne (EPFL)
School of Computer and Communication Sciences
CH-1015 Lausanne, Switzerland
{gil.regev, alain.wegmann}@epfl.ch
[2]IbisSoft, Stockholm, Sweden
ilia@ibissoft.se

Abstract. Enterprise survival is about maintaining an identity that is separate from other enterprises. We define flexibility as the ability to change without losing identity. The identity of an enterprise can be analyzed as a set of norms and beliefs about these norms held by its stakeholders, such as customers, employees, suppliers and investors. Business processes and their support systems maintain invariants that are the result of compromises between the often conflicting norms and beliefs of these stakeholders. We formalize these invariants as values in a state space. Identifying a minimum set of invariants provides a basis for defining flexible processes and support systems. We illustrate the use of this framework with production business process support systems.

Keywords: Business Process, Flexibility, Stability, Invariant, Regulation.

1 Introduction

Enterprise survival often hinges on the fit between the enterprise's position in its environment on the one hand and its business processes and their supporting systems on the other. Flexibility is an essential property for the maintenance of this fit in changing environments (Knoll and Jarvenpaa 1994). Changes within the enterprise and in the enterprise's environment often introduce misfits between the enterprise and its environment. Business processes and Business Process Support systems (BPS) need to be flexible so that changes can be made to them to correct these misfits.

[*] The work of the second author was supported by the Swedish Agency for Innovation Systems (Vinnova) under the grant for a project on Integration of Business Process Support with Knowledge Management. We would also like to thank the anonymous reviewers who helped us to improve the structure and the language of this article.

Flexibility is usually thought of as the capability to change. It may, however, be useful to analyze flexibility from the opposite dimension of stability. From this perspective we argue that to change something, that something first needs to exist, i.e. without prior existence, change is not possible. Furthermore, to exist, that something needs to maintain a set of features that is different from all other things in its environment. This set of features enables observers of the thing to define its identity. For example, for an enterprise to exist, it has to have some features that are different from other enterprises. These features constitute the enterprise's identity in the eyes of its stakeholders. These features need to remain somewhat stable in order for the enterprise to maintain its identity. From this point of view, existence is about non-change, i.e. stability. To maintain these features stable, in a changing environment, the enterprise needs to continually adapt. Hence, there is no existence without change. Flexibility is therefore the ability to continuously balance between change and stability without losing identity.

To maintain its identity an enterprise manages relationships with stakeholders. Business processes are the methods used by the enterprise to manage these relationships. Business processes often represent a compromise between the conflicting norms of the different stakeholders of the enterprise. Compromises are needed, for example, in order to satisfy the differing expectations of customers, investors, suppliers, employees, and government regulators at the same time. In a business process point of view, we model the compromises maintained by business processes as invariants. These invariants represent what is important to maintain unchanged. We model an invariant as a subset of states in a state space. By defining a minimal set of invariants that satisfies the stakeholders, we can explore different possibilities for changing the business process and maintain its essential properties unchanged. We apply this analysis to three levels, namely the organizational level at which business process types are defined, the operational level at which business process instances are executed, and the BPS system level at which requirements for the support system are defined.

We begin this paper by exploring the issue of flexibility and the difficulties of providing flexibility in business processes from a theoretical point of view (Section 2). We continue with exploring ways for defining invariants in a more formal way (Section 3). To this end, we first describe concrete examples of business processes analyzed from the point of view of the invariants they maintain in a state space. We then propose a set of requirements that could be considered as an invariant for BPS systems. Next, we overview related research and discuss in what way it differs from what is proposed in this paper (Section 4). Finally, we give a brief summary of the main ideas of the paper (setion5).

2 Flexibility

In this section we explore, from a theoretical point of view, the issue of flexibility of business processes and their support systems. We begin by analyzing the relationship between flexibility and the maintenance of identity in general. We then discuss organizational flexibility, followed by business process and BPS system flexibility.

Flexibility and Identity

To understand flexibility, we take as a point of departure a dictionary definition of the term flexible (Merriam-Webster 2005):

1: capable of being flexed: Pliant
2: yielding to influence: Tractable
3: characterized by a ready capability to adapt to new, different, or changing requirements

These definitions suggest that something that is flexible, though yielding to influence must still resist this influence. It can be argued that most, if not all, of the things we see exist because they resist change to some extent. An enterprise, for example, must to some extent resist external influences such as demands by clients to provide better, less expensive service, price hike demands by suppliers, and regulators demands for stiffer regulations etc. Yielding immediately to all influences will tear up the enterprise quite rapidly.

From a state space perspective, the states occupied by some object are a subset of all possible states. This restriction to a subset of states enables observers to identify an object as separate from its environment. A flexible object can occupy a different set of states than its original subset of states. This second set is also a subset of all possible states. Furthermore, in order to be defined by an observer as “the same object” the second set of states should not differ too much from the first one. In other words, what we call flexible is neither so rigid as not to accept change at all nor so changeable that it falls into pieces when we attempt to change it. Flexibility, therefore, is the maintenance of a subset of permissible states in the face of change. This subset of states is often called structure.

Structure refers to recurrent patterns of the behavior of the system (the different states/actions and the relation between them) and to the construction (Dietz 2004) of the system (the sub-systems and their interrelations). As structure must be somewhat stable, it is by definition averse to change. This is both a blessing and a curse: A blessing as long as the conditions do not change and therefore as long as change is not necessary; A curse when change is necessary (Weinberg and Weinberg 1988).

Maintaining flexibility, i.e. both stability and the capability to change, requires an optimal structure. It should not be too complicated so that it cannot be changed when needed, nor too simple with respect to the challenges faced by the organization (Weinberg and Weinberg 1988). In other words, for the structure to be flexible it needs to not be too stable (i.e. rigid) nor too unstable (and lose its identity).

Organizational Flexibility and Identity

The structure of an enterprise (both behavioral and constructional) can be studied as a set of norms. A *norm* can be defined as a set of states that do not differ significantly from one another, i.e., a feature that remains relatively stable within some timeframe, defined by an observer. A norm, as viewed from the point of view of an observer, is usually called a *belief* (Regev and Wegmann 2005). Norms and beliefs interact in sophisticated ways; see for example, (Vickers 1987). For the purpose of this paper, we simply note that in order to change the norms of an enterprise, the beliefs of its stakeholders (its observers) need to change as well. For example, before a business process is changed, it is necessary to convince stakeholders that the current process is either faulty, or that it can be improved in some way (lower cost, more output, better conformance to government regulations etc). Without the stakeholders' acceptance, no process change is likely to occur. Hence, the people responsible for a process need to change their beliefs about the current process and the potential benefits of a process change.

The identity of an enterprise, for a given stakeholder, consists of a set of beliefs that the stakeholder has about this enterprise (Regev and Wegmann 2005). As enterprises have several stakeholders, some of which have conflicting interests in it, the enterprise needs to maintain multiple identities. For example, customers and suppliers of an enterprise have a different set of beliefs about the enterprise from those of the enterprise's employees. Employees are by definition more aware of internal norms of the enterprise. Customers and suppliers will tend to better perceive the norms that the enterprise exposes to the outside world.

Flexibility can be defined as change that may be made to a set of norms in a given amount of time, without affecting the beliefs that form the identity of the enterprise for a given observer. Therefore, to define what needs to change, we first need to identify what the identity of the enterprise is for its essential stakeholders. We can then define whether an envisioned change modifies this identity. It is then possible to decide whether to modify this identity or not. From a state space perspective, this requires the identification of the subset of states that should not change based on the viewpoints of a set of essential stakeholders. We call this subset of states, an *invariant*.

Business Process Flexibility

The commonly accepted definition of a business process is a partially ordered set of actions performed to reach a well-defined goal. The goal of a business process has traditionally been considered to be the value provided to the beneficiary of the process (e.g. customer), see for example (Hammer and Champy 1993). With this definition in hand it seems that a business process is flexible if it is able to absorb change without changing its goal and/or if the goal can be changed with or without changing the set of actions, as well as their order. However, there are usually more stringent limits on this flexibility.

To better understand these limits, we use the regulation-based view proposed in (Regev 2003) and (Regev et al. 2005). In this view a business process is *a set of interrelated actions that regulate a set of relationships among stakeholders*. Based on the discussion above, a business process can be seen as a behavioral norm (the partially ordered set of actions) of an enterprise. It is designed in order to maintain another set of norms, *the invariant*, unchanged.

This definition has several advantages. First, it shows that several stakeholders, not only the customer, have an interest in a process. Second, using the term regulation shows that business processes are designed to maintain some equilibrium between the conflicting requirements of the stakeholders. This equilibrium, the invariant, often takes the form of a compromise, or an accommodation according to Checkland and Scholes (1990) between these conflicting requirements. The business process is designed so that this equilibrium is maintained even when occasionally the stakeholders attempt to change this point of equilibrium. As discussed in (Regev et al. 2005) some of the actions of a business process can be understood as regulative actions designed to prevent stakeholders from "abusing" the business process, i.e. from modifying the carefully constructed equilibrium. This structure obviously is somewhat immune to change and therefore has limited flexibility.

For example, the process defined in order to sell some product to a customer (called here the Sales/Collection process) is designed in most commercial enterprises as a set of actions that ensure that the product required by the customer is delivered to the customer in exchange for payment. The payment is calculated to ensure that the stakeholders of the enterprise are all paid, i.e. the suppliers, employees, tax collector, investors etc. This payment is needed so that they maintain their stake in the enterprise within a given norm, e.g. that suppliers continue to provide raw material to the enterprise. Similarly, products shipped need to match customer orders so that customers maintain their stake in the enterprise within a given norm, e.g. so that they order other products from the same enterprise in the future.

In this example, there is obviously a tension between the customer's desire to get the product as inexpensive as possible and the suppliers', investors' and the organization's employees' desire to be paid as much as possible. Government regulators also place requirements on the quality of the manufactured product and the taxes to be collected. The process has built-in actions to prevent any of these variables deviating from the prescribed equilibrium. Actions such as quality control, batch numbering and tax calculation are performed in order to satisfy government regulators. These regulators are themselves influenced by customers and businesses in an endless cycle of norm setting as described by Vickers (1987).

Hence, from a regulation point of view, we can notice that this process is designed to ensure that a compromise, a set of interdependent norms of the enterprise and its stakeholders, is maintained unchanged. In this view, the goal of the process is the satisfaction of the process client's norms, for example, the delivery of the required product. It is a useful simplification to see a process as satisfying this goal with the norms, of the other stakeholders, imposing constraints on the satisfaction of this goal.

Because it needs to maintain unchanged a disparate set of norms, a business process has a limited amount of flexibility. The Sales/Collection process, for example, cannot be changed so that payment is not required anymore, or that the product is not delivered, or that the actions defined so that the correct product is delivered to the customer are removed. These and many other changes will make the Sales/Collection process ineffective in maintaining the norms of the enterprise and its stakeholders.

The constraints imposed on the process by each stakeholder's norms render the process rather inflexible, but at the same time the participation of the stakeholders in the process, as well as their norms, render the process possible and flexible in another dimension. Indeed, the stakeholders' norms provide the foundations on which the process is built. For example, a Sales/Collection process cannot exist without suppliers, investors, employees etc. Investors enable flexibility by providing capital when needed to the organization, but they also require a return on investment. This limits the flexibility of the business process with respect to discounts that can be offered to customers.

Other reasons for the inflexibility of business processes are the need for stability of the organization and its stakeholders. People in organizations need some stability after having learned a new process. Processes that change too often (i.e. faster than their stakeholders can accept) are ineffective, no matter what useful innovations they introduce. Organizations need to capitalize on a process, i.e. get a return on this investment. A proposal to change a recently changed process is likely to be met with resistance from finance departments intent on reaping the rewards from the recent change. Change is sometimes painful for people and almost always expensive for the organization. People in organizations do not always realize that a change is possible. Hence, an organization that goes through a business process change usually attempts to avoid going through another process change for some time.

Because it is difficult and even dangerous to change business processes continually, it is sometime necessary to change just the occurrence of one execution of a business process. In other words, it is necessary to occasionally deviate from the norm governing the business process. This kind of change is necessary when, for example, a customer requests some special treatment when purchasing a product. The Sales/Collection process instance for that customer may be modified without altering the general process type.

We therefore distinguish between organizational change, i.e. changes that can be made to a business process type (the norm that describes the actions to be executed, their sequence and the expected outcome of the process) and changes to a business process instance without making durable changes to the process type; see also (Soffer 2004).

Business Process Support systems need to be flexible enough so that they can accept both changes to business process types and business process instances at a reasonable cost. Ideally, business process instance changes should have minimal cost.

3 Towards a Formal Definition of Invariants

As follows from the previous sections, flexibility can be thought of as the ability to change without losing "identity". Therefore, defining a degree of flexibility can be achieved more easily through stating what should not be changed rather than what can be changed. Identity, as we have seen, can be studied as a set of norms, or invariants. The question arises about how to define an invariant that should be preserved at any cost in some strict manner. The question is not easy to answer, as the method for defining identity should be quite "flexible" in the sense that it should allow various degrees of flexibility to be defined.

As we distinguish a number of different types of flexibility in connection with business processes and support systems, it is difficult to expect that invariants for different flexibility types could be defined in the same way. In the following subsections, we will try to develop a path for defining invariants for two kinds of flexibility, namely, process instance flexibility, and process support system flexibility. We do not suggest that these approaches are the only ones that can be used for specifying these particular invariants. In addition, we do not intend to fully develop a formal representation of such invariants in this paper. Our goal is much simpler, specifically, to demonstrate that there are some ways for defining invariants. And therefore the approach for specifying a degree of flexibility through invariants deserves to be taken into consideration.

Invariants for Process Types

As was stated above, the process instance flexibility is the possibility for a process instance to deviate from the standard established for a particular process type. In the terms of the identity theory, we need to define an invariant for a process type that should be preserved for any process instance that belongs to this type.

Let us start with an informal discussion to demonstrate what kind of deviations should/could be “permitted.” We will compare two standard processes that can be found in all industrial organizations, see (Denna et al. 1995) namely, Acquisition/Payment and Sales/Collection. Acquisition/Payment is aimed at buying raw materials from suppliers, and Sales/Collection is aimed at selling goods to customers and receiving money in return. The instances of these two process types should be kept separate. Even when we allow the maximum possible flexibility, an instance of the Sales/Collection type, for example, should not begin to look like an instance of the Acquisition/Payment process.

Let us discuss in what way we can define an invariant of the Sales/Collection process so that it will differ from Acquisition/Payment. For example, we can try to define the process in terms of actions (steps) to be completed. Clearly, the Sales/Collection process includes actions such as *package merchandise*, *ship it*, and *receive payments*. However, these actions may be found in some instances of the Acquisition/Payment process as well. For example, if the enterprise receives wrong goods from a supplier, the enterprise would need to package and ship them back, as well as to obtain a refund (return payment) from the supplier.

As the kind of actions defined in a process is not enough, maybe the order of the actions will help us to define an invariant. Unfortunately, if we want to be able to define a high degree of flexibility, this will not help as the order is difficult to fix even inside one and the same process type. For example, for the Sales/ Collection process with a high degree of flexibility both orders

- ship merchandise
- receive payment

and

- receive payment
- ship merchandise

are quite thinkable. The first is normal for sales to known and trusted customers. The second one is normal for sales to unknown or non-trusted customers.

The example above shows that defining an invariant for a process type is not a trivial matter, the task requires choosing a proper theoretical view on the concept of business process: the one that is, as much as possible, independent from the notions of action and action order. According to the most general definition, a business process is a partially ordered set of actions performed to reach a well-defined goal. There are several approaches to the formalization of this concept, the workflow thinking, which is based on the action flow, being the most widespread. Our classification that distinguishes four different views on business processes is given in (Bider 2005), and, for lack of space, it is not repeated in this paper. The formal definition of an invariant will certainly depend on the view chosen, thus we need to choose a particular view before proceeding with formalizing the idea of invariants. In this paper, we use the state-oriented view on business processes (Khomyakov and Bider 2000, Bider 2002), as it is less dependent on actions and their ordering than the other views known to us.

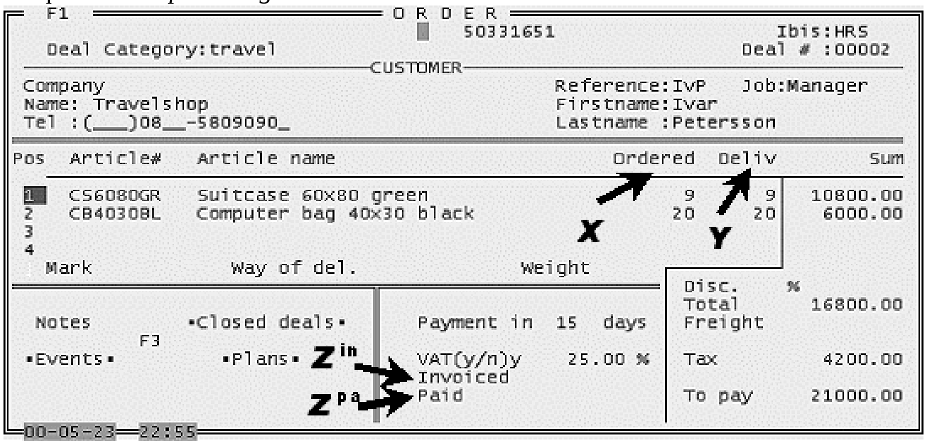
The main concept of the state-oriented view is a *state* of the process instance that can be defined as a position in some state space. A state space is considered multidimensional, where each dimension represents some important parameter (and its possible values) of the business process. Each point in the state space represents a possible result of the execution of a process instance. If we add the time axis to the state space, then a trajectory (curve) in the space-time will represent a possible execution of a process instance in time. A process type is defined as a subset of allowed trajectories in space-time. Thus, in terms of the state-oriented view, the invariant for the process type can be defined as an intension of such a subset, i.e., a logical formula that differentiates the trajectories that belong to the set from those that do not belong to it.

Before further outlining possible ways of defining invariants, let us illustrate the basic notions of the state-oriented thinking on the already introduced example of the Sales/Collection process. Figure 1 represents a possible structure of a state space for Sales/Collection. This figure is a screen capture from a production system built by IbisSoft in the late 1980s based on the state-oriented thinking (see (Andersson et al 2005, Bider 1997, Bider 2002) and an explanatory note at the end of this section). The following main numerical dimensions can be distinguished for the Sales/Collection process based on Figure 1:

- There are a number of pairs of product-related dimensions <*ordered*, *delivered*>, one pair for each product being sold. The first dimension represents the number of ordered items of a particular product. The second one represents the number of already delivered items of this product. The number of such pairs of dimensions is not fixed but is less than or equal to the size of the company's product assortment. Denote product dimensions as $X^1,\ldots,X^n$ (ordered) and $Y^1,\ldots,Y^n$ (delivered), each index representing a product category from the assortment. In Figure 1, dimensions $X^1,\ldots, X^n$ are represented by column *Ordered* (see arrow X), dimensions $Y^1,\ldots, Y^n$ are represented by column *Deliv* (see arrow Y).

- In addition, there are two numeric dimensions concerning payment: *invoiced* (amount of money invoiced) and *paid* (amount of money already received from the customer). These dimensions are denoted as Z^{in}, and Z^{pa}. In Figure 1, dimension Z^{in} is represented by the field labeled *Invoiced* (see arrow Z^{in}), and Z^{pa} is represented by the field labeled *Paid* (see arrow Z^{pa}).

In the example in Figure 1, the purchase order comprises 9 suitcases and 20 computer bags. For the suitcases, X^{i} is equal to 9, Y^{i} is also equal to 9 (all ordered suitcases have already been delivered), and index *i* is equal to *Suitcase*. For the computer bags, X^{i} is equal to 20, Y^{i} is also equal to 20 (all ordered bags have already been delivered), and *i* is equal to *Computer bags*.



```
 F1                              O R D E R
                                     50331651                     Ibis:HRS
 Deal Category:travel                                           Deal # :00002
                              CUSTOMER
Company                                        Reference:IvP    Job:Manager
Name: Travelshop                               Firstname:Ivar
Tel :(___)08__-5809090_                        Lastname :Petersson

Pos  Article#    Article name                    Ordered  Deliv          Sum
1    CS6080GR    Suitcase 60x80 green                  9      9     10800.00
2    CB4030BL    Computer bag 40x30 black             20     20      6000.00
3
4
 Mark              Way of del.            Weight
                                                       Disc.     %
 Notes        •Closed deals•    Payment in  15  days   Total        16800.00
            F3                                         Freight
•Events•        •Plans•         VAT(y/n)y   25.00 %    Tax           4200.00
                                Invoiced
                                Paid                   To pay       21000.00
00-05-23  22:55
```

Figure 1 State space of Sales/Collection process represented as a screen, from (Khomyakov and Bider 2000).

With the dimensions set as above we can set at least one commonsense restriction on the set of trajectories of the Sales/collection process. This restriction is connected to the “static” goal of the Sales/Collection process from the point of view of the selling enterprise, and it can be expressed as follows:

“An instance of Sales/Collection is considered to be properly finished when all ordered goods have been delivered to the customer and the agreed sum of money has been collected from the customer.”

We call such goal *static* because it does not set any restrictions on how the finished state of the process can be achieved. For example, it does not state how the goods should be (have been) delivered: in one step or in several steps, the next working day or over a year. Neither does it state how the money should be (have been) received: before delivering the goods, in time or after a reminder.

Formally, the static goal defined above can be expressed in the following way. Each trajectory of the Sales/Collection process should end at the surface in the state space defined by the equations:

(a) $x_n = y_n$, $z_{in} = k_1 x_1 + \dots + k_n x_n + t + f$, $z_{pa} = z_{in}$, where x_n – is a position of the process instance along axis $X^{n,}$, y_n – is a position of the process instance along axis $Y^{n,}$, z_{in} – is a position of the process instance along axis $Z^{in,}$, $k_1, \dots, k_n$ represent prices of the products ordered, t represents *VAT*, f represents *freight*, and z_{pa} – is a position of the process instance along axis Z^{pa}

Note that the expression (a) includes VAT collection as a mandatory part of the product price. This shows that the Sales/Collection process type is designed in such a way as to satisfy several stakeholders, including the government, see discussion in Section 2, and (Regev et al. 2005)

Does expression (a) set enough restrictions on the trajectories of Sales/Collection to be used as a "minimum" invariant of this process type? For example, can this formula differentiate Sales/Collection from Acquisition/Payment? The static goal of the Acquisition/Payment process is receiving all ordered goods and paying the agreed price. Actually, formula (a) may be used to represent the Acquisition/Payment process if interpreted from the point of view of the buying company. If we consider negative values for *ordered* and *delivered* as goods being bought by the enterprise instead of sold, and negative numbers for *invoiced* and *paid* as money paid by the enterprise, the goal of the Acquisition/Payment process can also be expressed as the surface defined by the same formula (a) above. If we abstract the nature of the exchanged goods, the only difference between the processes is that Sales/Collection requires non-negative values for $X^1, \dots, X^n$, whereas Acquisition/Payment requires non-positive values for $X^1, \dots, X^n$.

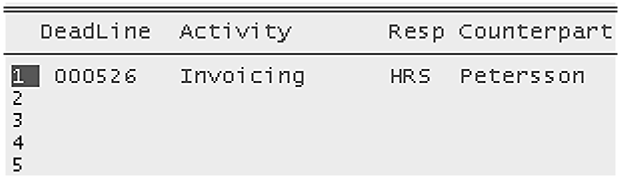
As follows from the above deliberation, defining a static goal for the process helps at least to differentiate Sales/Collection from Acquisition/Payment. Is a static goal definition enough to serve as a process type invariant? Not quite, because a statement of a static goal does not guarantee that a particular instance will try to reach the goal, and, even less, that the goal will be reached. We need some constraints on the trajectory itself. Can we define these constraints by limiting the points through which the instance trajectory can travel? Can we, for example, state that for the Sales/Collection process, x_n (ordered) should always be greater or equal to y_n (delivered)?

We could impose strict conditions on the trajectory only in the case were we have full control over the process. This is never the case in the business reality. Influence of external forces (environment) can make it possible for a process instance to land in virtually any part of the state space. Consider, for example, that due to a human error 10 green suitcases, instead of 9, were delivered in the process instance from Figure 1. If the error is discovered, the number of delivered items (10) becomes more than the number of ordered items (9). Some error correcting mechanism needs to be built in the process in order for each instance having a chance to reach the static goal. Such mechanism is needed for business processes to fulfil their regulative roles as discussed in (Regev et al. 2005).

How can we express this correcting mechanism in the terms of invariants? Consider an analogy from the domain of physical process control. Suppose we believe we have navigated a ship to a certain place in the ocean, and due to the wind and currents have actually missed the place. The only thing to do in this situation is to turn around and re-set the course. Thus, the limitation on the valid trajectory is not about where you can get to, e.g. by chance, but about where you aim to go, i.e. your goal. Always aiming towards the goal constitutes an “error-correcting mechanism” of navigating a ship to a certain point.

Applying the logic of physical process control to business processes, we need to require that the process instance be always aimed at its goal. To express this fact for physical processes, the derivatives are used when the process trajectory is described in the form of differential equations. The latter formalism is not particularly suitable for the business processes. What can we use instead to specify the direction and speed of movement of a business process instance?

In the state-oriented view on business processes (Khomyakov and Bider 2000), the idea of a process plan is used to express the direction and speed of movement of a process instance. The plan is a list of actions to be executed to move the process instance forward. The plan defines actions that should be completed with possible parameters, resources assigned, and time set for completing the execution, see example in Figure 1. The action defines along which axes the plan will move the process instance, parameters may define for how long and how fast (using deadlines, for example).



	DeadLine	Activity	Resp	Counterpart
1	000526	Invoicing	HRS	Petersson
2				
3				
4				
5				

Figure 2 A process plan that complements the state from Figure 1

Let us assign to each action (allowed in the frame of a given process type) axes of process state and direction of movement along them. For example:

- shipping to customer– positive movement alongside axes Y^1,…, Y^n
- receiving a customer return – negative movement alongside axes Y^1,…, Y^n
- invoicing customer – positive movement alongside axis Z^{in}
- getting paid by customer– positive movement alongside Z^{pa} axis
- crediting customer – negative movement alongside Z^{in} axis
- returning money – negative movement alongside Z^{pa} axis

After that, we can formulate two requirements that would be reasonable to impose on the plan of any business process instance:

1. Actions in the plan should always represent movements from the current point in the state space in the direction to the goal.
2. The plan should include at least one action.

These requirements ensure constant movement towards the goal. Note that both requirements are fully satisfied for the plan in Figure 2. Together with the static goal and actions specification these requirements constitute a “most flexible” invariant of a process type. Of course, in real life more strict rules can be applied like “never ship until money is received”. This can be included in the planning rules as discussed in (Khomyakov and Bider 2000).

The definition of an invariant as suggested above includes two limits: lower (minimum), and upper (maximum) limits. The minimum requires that an invariant consist of having a goal plus being always directed to the goal. The maximum limit requires that constraints on the trajectory should not be defined in terms of positions in the state space, but in the terms of movements.

Movement constraints can be expressed as restrictions on the direction of movement (planned actions), speed of movement (deadlines in the plan), or as general restrictions on the whole trajectory. An example of the later kind is a time limit for finishing a process instance. This can be a general limit for all instances (e.g., next day delivery), or a limit negotiated for each instance with a customer. To have the latter for the Sales/Collection process on Figure 2, an additional parameter, *delivery date*, needs to be introduced. Having such a parameter instead of a general rule gives more flexibility, as in the case of a serious trouble, the delivery date can be renegotiated with the customer. Note that introducing a new parameter changes the shape of the state-space. We consider the shape of the state-space to be a characteristic of the process type, rather than a process instance. Changes in a process type, such as adding or removing a dimension, fall outside the scope of this section.

Breaking the rule of allowing only movement constraints will lead to either too much or too little flexibility, or, most probably, to both. Too much, for example, means allowing actions that are not directed towards the goal to be completed. Too little, for example, means allowing a process instance to be stuck in a dead-end point of the state-space.

Note: Figures 1, 2 above, and 3 below represent screenshots from a business process support system called *DealDriver* developed in 1989-1990. The case represented in the screenshots is artificial but, in essence, it does not differ from any real case recorded in the system logs. For more details see (Andersson et al. 2005, Bider 1997, Bider 2002).

Architectural Invariants for Business Process Support Systems

As was stated in the previous section, we consider system flexibility as the ability to acquire new or changed functionality with a minimum cost. In this section, we considered a special kind of invariant that concerns this kind of flexibility, which we call architectural invariants. An architectural invariant represents a number of basic concepts that should be implemented in a Business Process Support system (BPS). Although we call this kind of invariant architectural, we do not refer to any technical means of implementing these concepts. In an architectural invariant, concepts are defined at a high abstraction level, and different systems can implement these concepts with the help of different technical and user-interface means.

For architectural invariants, we also introduce the notions of minimum, and maximum invariants, but in a slightly different sense than in the previous subsection. A minimum invariant lists the minimum number of basic concepts that should be implemented in the system in order for it to have the right to be called a BPS. A maximum invariant lists the maximum number of basic concepts that can be implemented in the system in order for it to still be called a BPS. A maximum invariant does not prohibit adding a new functionality, it only requires that all new functionalities should be represented in terms of allowed basic concepts.

To be able to discuss architectural invariants of BPS we need to agree on what a business process support system is. In general, we can regard any computer system that helps to run a business as a BPS system. With such a definition, an accounting system, or even a word processor can be viewed as a kind of BPS system. For the sake of this discussion, however, we would like to have a more narrow definition. After that, we can define architectural invariants that will help to differentiate BPS systems from any other business applications, including, an accounting system and word processor. We define a BPS system from two perspectives, a business perspective, and theoretical one. Both perspectives represent the same belief on what a BPS system is, though, in different terms.

From the business perspective, we consider a BPS to be a system that facilitates business process-orientation as a way of working. The latter means genuine cooperation between all process participants independently of which department they belong to, and whether a particular process instance follows the standard pattern or deviates from it. It also means motivated involvement of process participants who understand their own roles in the process and the roles of others, including management. Business process orientation as a way of working also means that the experience gathered from previously completed processes is directly used in operational practice. To facilitate the process-oriented way of working, a BPS system should provide the following functions that are specific for this kind of business support systems:

1. Give to each process participant easy access to the state of affairs in any particular process instance. This includes information on what has been achieved so far, how it has been achieved, and what will be done in the nearest future.
2. Give to each process participant easy access to all process instances he/she is participating in, including information on what he/she is supposed to do in the frame of each process instance and when.
3. Provide participants with effective communication channels along the process instance lines.
4. Provide easy access to the organization’s experience, e.g., already finished processes, so that it can be analyzed, and participants can learn by example.

From the theoretical perspective, we consider a BPS to be a system that helps people to ensure that business process instances comply with a set of invariants defined by a given business process type. This means navigating business process instances towards their goals while complying with the relevant norms of the business process stakeholders. As we deal with the general issues of flexibility, we require that flexibility with respect to process instances is supported by the system, i.e. the system helps the stakeholders to understand which deviations from the invariants are reasonable.

We believe that, at a minimum, we need three functions:

1. The possibility to point out where we are.
2. The possibility to lay our course forward to the goal from current position.
3. The possibility to register our movements and track them backward if needed.

The first two requirements from the list above logically follow from the discussion in the previous section. The last requirement needs some explanation. Tracking satisfies various needs. For example, it gives us material for analysis, and it can be used for investigating process performance, behavior of external environment, etc. However, it is also needed for the sake of operational control over process instances, which we consider to be the primary objective of a BPS. One such need is connected to the problem of identification of the current position in the state space, which can very seldom be measured exactly. When we ship goods in an instance of Sales/Collection, we normally consider them to be delivered and change our position along the *Y*-axes correspondingly. If we have a reliable delivery channel, this approximation will work in the majority of cases. However, if the customer contacts us and complains that some goods are missing, we need to track the process back to find out which of the deliveries made went wrong and when, as well as where to find the missing goods. Also regulation bodies (e.g. government) often require backtracking of business processes for auditing purposes or when problems such as accidents occur.

In terms of the state oriented view of business processes, the above list can be formulated as requirements for implementing a number of basic concepts:

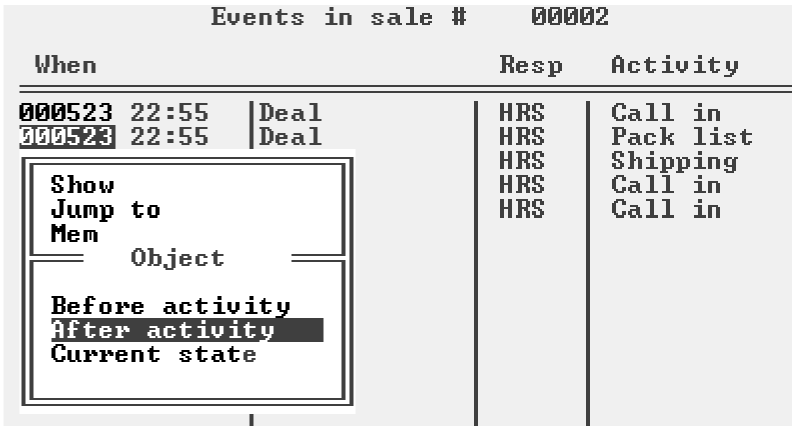
1. Having a representation of the state space and a position in it of any instance at any point of time. This can be done, for example, in the form of a screen similar to the one presented in Figure 1.
2. The possibility to plan each process instance independently from all others. This is needed to be able to lay out a course that keeps a process instance in the frame defined by the process type invariant. A plan may have a form as in Figure 2.
3. The possibility to register all events, e.g., actions executed in the frame of a given process instance. In addition, we need means to browse positions in the state space after each registered event. Registered events may, for example, be represented as a list of already completed actions, see Figure 3. A position in the state space in the past can be shown in the same way as it is done for the current state, e.g., as in Figure 1. Browsing to this point can be done, for example from a menu on the event list as shown in Figure 3.

The above requirements can be thought of as the minimum possible invariant of a BPS system. Implementing only this minimum in a BPS leaves a considerable amount of manual work, e.g.:

- Removing completed actions from the plan
- Registering a completed action in the event list, and calculating and registering the results achieved by this action in the state space

- Laying a new course by planning additional actions
- Manually executing actions outside of the BPS system

Implementing only the minimum also means that the responsibility of keeping a process instance inside the frame set by an invariant is still left to the people participating in the process. They need to impose the restriction based on explicit or implicit rules, manuals etc.



```
                 Events in sale #    00002

 When                                 Resp   Activity
000523 22:55  |Deal                  | HRS  | Call in
000523 22:55  |Deal                  | HRS  | Pack list
                                       HRS    Shipping
                                       HRS    Call in
                                       HRS    Call in
```

Figure 3 Events from the process instance in Figure 1

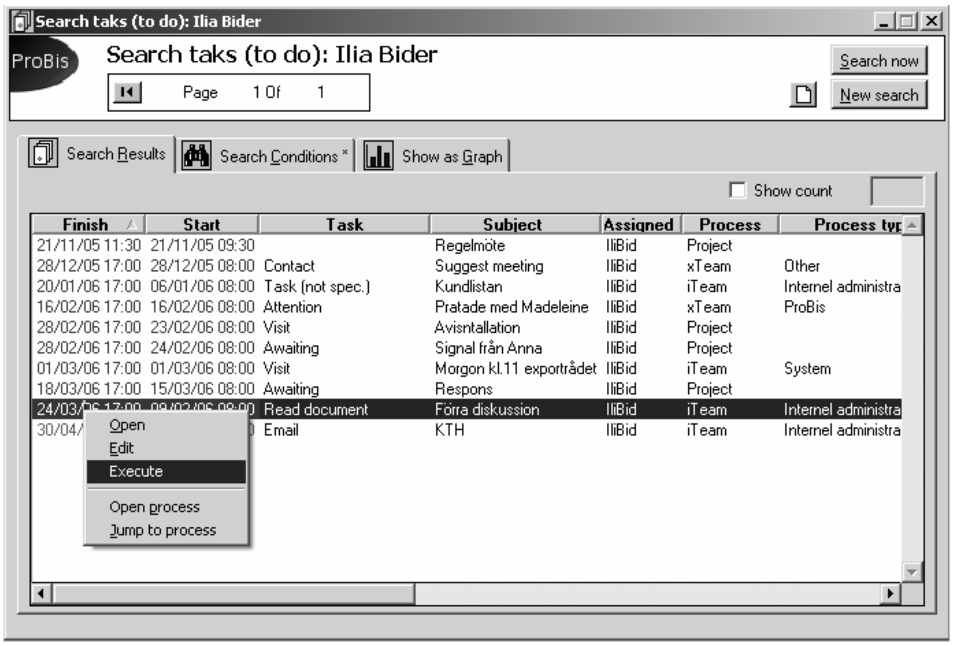
A minimum BPS can be compared with a simple word-processor (Notepad, for example) that allows writing any text but does not provide any means for checking spelling and grammar. Let us consider what kind of extra help can be obtained from a system for navigation purposes. The following assistance in navigation is thinkable:

1. Assistance in registering events and movements in the state space. This, for example, can be done when an action in the plan is chosen to be reported as executed. It can then be automatically removed from the plan, and inserted into the event list. In addition, some help can be provided when calculating a new position in the state space.
2. Automated course laying – assistance in planning new actions and assigning resources to them. This feature should assure that any instance remains in the frame of the invariant set for the given process type. According to (Khomyakov and Bider 2000, Bider 2002), three types of planning rules can be differentiated: obligations, prohibitions, and recommendations. The first two types of rules help to impose the invariant. The third one is meant to guide the people to keep any instance as close as possible to the most desirable trajectory.
3. Assistance in executing actions outside of the BPS system; for example, automatically shipping a software product via email.

In some circumstances all three functions above can be fully automated, which makes it possible to put a process instance under autopilot for some time. As soon as conditions change, and human assistance is required, the system can request such assistance by planning an action in which a human being should participate.

Consider adding the concepts of three “assisting” functions above to the minimum invariant suggested earlier to define the maximum invariant for BPS. In this case, no features can be added to the system unless they can be expressed in the terms of the six concepts included in the maximum invariant. How much flexibility will we be left with then? Actually, with quite a lot, and we will show this with two examples below.

Personal Calendar. An ultimate goal of a BPS system is helping people to run their business, be it official, company or private. Having a personal calendar that reminds people of the tasks to be completed would be quite a natural feature. However, at first glance, the maximum invariant we defined does not allow the introduction of personal calendars. This is true if we insist on a direct interpretation of such a feature. Nevertheless, indirectly, personal calendars already exist among the concepts of our invariant. A BPS system should support the planning of actions in the frame of various process instances and the assignment of resources to them, including human resources. A view of all actions planned for a particular person in all currently running process instances, in fact, constitutes a personal calendar for this person, see, for example Figure 4. This view provides the second function defined in the business perspective discussed above.



Finish	Start	Task	Subject	Assigned	Process	Process typ
21/11/05 11:30	21/11/05 09:30		Regelmöte	IliBid	Project	
28/12/05 17:00	28/12/05 08:00	Contact	Suggest meeting	IliBid	xTeam	Other
20/01/06 17:00	06/01/06 08:00	Task (not spec.)	Kundlistan	IliBid	iTeam	Internel administra
16/02/06 17:00	16/02/06 08:00	Attention	Pratade med Madeleine	IliBid	xTeam	ProBis
28/02/06 17:00	23/02/06 08:00	Visit	Avisntallation	IliBid	Project	
28/02/06 17:00	24/02/06 08:00	Awaiting	Signal från Anna	IliBid	Project	
01/03/06 17:00	01/03/06 08:00	Visit	Morgon kl.11 exportrådet	IliBid	iTeam	System
18/03/06 17:00	15/03/06 08:00	Awaiting	Respons	IliBid	Project	
24/03/06 17:00	[illegible]	Read document	Förra diskussion	IliBid	iTeam	Internel administra
30/04/	[illegible]	Email	KTH	IliBid	iTeam	Internel administra

Figure 4 Personal calendar seen as a special view on business process instances

Internal Communication along Process Instances. For many business processes, achieving the goal of a process instance requires the participation of several people. It is natural to expect that a BPS should support their cooperation and communication. Should we introduce a separate communication channel in the system not prescribed by the maximum invariant above? Actually, a BPS invariant indirectly contains a concept of one type of communication channels, i.e. the one that concerns asynchronous communication along the business process instances. As was stated in the invariant, the system should support planning and resource assignment. Let us allow planning "communication" actions, like *attention*, *help needed*, etc., to our colleagues. They will receive these actions in their personal calendars (see Figure 4) in the same way as all other actions, such as *shipping*. Moreover, each communication action will be connected to a particular process instance, which allows us to be concise when writing our messages. Placing a message in the frame of a particular process instance gives the addressee all information on the current state, history and future of this instance. Therefore, we do not need to repeat it in the message itself. Planning communication actions constitutes an effective communication channel along the process instance lines. See function 3 in the business perspective discussion above.

As we demonstrated in the examples above, imposing the BPS invariant described in this section leaves enough flexibility as far as adding new functionality is concerned. The question may arise about why we need such an invariant in the first place. Why not allow adding functionality in any way the stakeholders want? The answer is simple, if we allow for an uncontrollable development of a fairly complex system such as a BPS, it soon will become an internal mess that will be difficult to maintain, support, and even understand from the users' point of view. In this case, we can expect loss of identity. Setting an invariant enforces the development of a stable core of code on which each new piece of functionality is based.

Architectural invariants represent the norms and beliefs of system architects and software engineers who belong to the stakeholders of any business activity heavily dependent on a software system. It is important to take these norms and beliefs into consideration just as one would for the norms and beliefs of any other stakeholder. Having an unstable system is as dangerous as having an inflexible (rigid) one. In a worst-case scenario, any of the extremes can lead to a company going out of business.

The six functions included in our BPS invariant were derived from the state-oriented view on business processes. Nevertheless, we believe that they will be the same even if we change the view used for building a BPS system, say to a workflow view. The technical architecture and user interface will change, but on the deeper abstract level, the functionality provided by the system should remain the same.

Note: Figure 4 represents a screenshot from our new BPS system, called *ProBis*, for more details see (Andersson et al. 2005).

4 Related Work

The business process movement emerged out of the need to overcome the rigidity that characterized so called traditional enterprises. Hence, most work done on flexibility assumes that it is about accepting and/or generating change. See for example (Hammer and Champy 1993) where radical change is advocated. It has been recognized that this focus on change is often non-beneficial for enterprises because it requires more change than what they can accommodate (Hammer 1996). Despite this focus on change, it is also recognized that early Workflow Management Systems (WFMS) lacked flexibility and therefore much research has been targeted into making these systems flexible (Ellis et al. 1995). Again, the focus is on change rather than stability. In the information systems field, too, research into flexibility assumes that it is about change only; see for example (Knoll and Jarvenpaa 1994). In these works, rigidity is often seen as negative. The focus is on maximum capability to change. Our work is different: our focus is on stability as a major component of flexibility. In this work, stability is seen not only as inevitable but as a necessary aspect of flexibility and survival. With our insistence on norms, our work does bear resemblance to Organizational Semiotics and its applications to business processes. See for example (Shishkov et al. 2002).

We would like to point out that there is a difference between our approach to business process instance flexibility and the way in which workflow-related research treats the flexibility issues. The most common way for introducing flexibility in workflow is by moving from the rigid predefined control flow to allowing some deviations (Blumenthal and Nutt 1995, Bogia 1995, van der Aalst 2001]. This, for example, can be achieved by permitting a process instance to deviate on particular paths. This way is quite appropriate when we want to localize flexibility to certain areas of the state space. Our approach is based on invariants and allows flexibility to be defined in more general terms. For example, the minimum invariant we defined is not localized and it allows finding a way out from practically any possible situation. We believe that our approach is more suitable for the processes that require substantial flexibility, for example, those that are connected with complex decision making.

It is also worthwhile to mention that recent trends in Software Engineering, e.g., agile and eXtreme Programming, e.g. (Beck 1999) mostly focus on change, and do not give enough attention to the stable structures behind the scenes. As follows from Section 3 of this paper, we focus on creating stable structures in software, as well as in business. This does not mean that we reject the ideas of agile development. We accept them with the addition that agile development of any complex system (e.g. BPS) is impossible without having or developing a stable structure, which can be expressed with the help of structural invariants.

5 Conclusions

Business process flexibility is most often seen from the point of view of maintaining the capability to change easily and quickly. In this paper we propose to view business process flexibility from the point of view of what needs to remain unchanged. We have identified the unchanging aspects of an enterprise as originating from the need to maintain a constant identity for a set of stakeholders. We analyze business process and support system flexibility on three levels. The uppermost level concerns the identity of the enterprise and results in possible changes to business process types. The second level concerns possible changes to business process instances. The third level concerns changes to the business process support systems. In each level, possible changes are dependent on the definition of the identity of that level by different stakeholders. We propose to view business processes as maintaining invariants in a state space. For a given business process, defining the minimal set of invariants that satisfies the stakeholders of the enterprise, enables many possible alternatives to be explored. This results in flexibility for both business processes and the associated requirements for their support system.

To investigate whether the idea of invariants can be used in practice, we undertook several steps in the direction of finding formal ways of defining invariants. We showed how Business Process and Business Process Support system invariants could be defined based on the state-oriented view on business processes. These first steps seem promising. However, it is too early to say how well our approach suits the task of defining invariants for practical purposes. For example, when defining business process invariants, we employed a dichotomy approach: the trajectory either belongs to a set or not. In practice, more granularity might be needed to differentiate the trajectories that are closer to a recommended standard from those that deviate substantially from the standard. In terms of our state-oriented model, the granularity can be achieved via the planning rules of type recommendations (Khomyakov and Bider 2000, Bider 2002). Currently, we are conducting the research aimed at the formal expression of recommendation and the ways of implementing them in a BPS system.

6 References

1. Andersson T., Bider I., Svensson R. 2005. Aligning people to business processes. Experience report. *Software Process: Improvement and Practice* 10(4): 403-413. DOI: 10.1002/spip.243.
2. Beck K. 1999. *Extreme programming explained: embrace change*. Addison-Wesley.
3. Bider I. 1997. Developing Tool Support for Process Oriented Management. *Data Base Management*. 26-01-30. Auerbach.
4. Bider I. 2002. *State-oriented business process modeling: principles, theory and practice*. PhD thesis, KTH (Royal Institute of Technology), Stockholm.
5. Bider I. 2005. Choosing Approach to Business Process Modeling. Practical Perspective". *Journal of Conceptual Modeling*, Issue 34. http://www.inconcept.com/jcm/January2005/IBider.html
6. Blumenthal R., Nutt G.J. 1995. Supporting Unstructured Workflow Activities in the Bramble ICN System. *Proceedings of the 1995 ACM Conference on Organizational Computing Systems (COOCS'95)*. Milpitas, CA. 130-137.
7. Bogia D.P., Kaplan S.M. 1995. Flexibility and Control for Dynamic Workflows in the WORLDS Environment. *Proceedings of the 1995 ACM Conference on Organizational Computing Systems (COOCS'95)*. Milpitas, CA. 148-159.
8. Checkland P., Scholes J. 1990. *Soft System Methodology in action.* Wiley: UK.
9. Denna E., Perry L., Jasperson J. 1995. Reengineering and REAL Business Process Modeling. Grover V., Kettinger W. Business Process Change: Reengineering Concepts. Methods and Technologies. Idea Group: UK.
10. Dietz J.L.G. 2004. Basic notions regarding business processes and supporting information systems. *Proceedings of the 5th BPMDS Workshop. Riga, Latvia.*

11. Ellis C., Keddara K., Rozenberg G. 1995. Dynamic change within workflow systems. *Proceedings of Conference on Organizational Computing Systems*. Milpitas, CA.
12. Hammer M., Champy J. 1993. *Reengineering the Corporation: A Manifesto for Business Revolution.* Nicholas Brealey: UK.
13. Hammer M. 1996. *Beyond Reengineering.* HarperCollins: NY.
14. Khomyakov M., Bider I. 2000 Achieving Workflow Flexibility through Taming the Chaos. *Proceedings of 6th international conference on object oriented information systems (OOIS 2000)*. Springer: 85-92.
15. Knoll K., Jarvenpaa S.L. 1994. Information technology alignment or “fit” in highly turbulent environments: the concept of flexibility. Proceedings of the 1994 computer personnel research conference on Reinventing IS. Alexandria, VA. ACM: 1 – 14.
16. Merriam-Webster Online http://m-w.com/. Accessed February 2005.
17. Regev G. 2003. *A Systemic Paradigm for Early IT System Requirements Based on Regulation Principles: The Lightswitch Approach*. Ph.D. Thesis. Ecole Polytechnique Fédérale de Lausanne (EPFL). Lausanne, Switzerland.
18. Regev G., Wegmann A. 2005. Where do Goals Come from: the Underlying Principles of Goal-Oriented Requirements Engineering. *Proceedings of the 13th IEEE International Requirements Engineering Conference*. Paris, France.
19. Regev G., Alexander I.F., Wegmann A. 2005. Modelling the regulative role of business processes with use and misuse cases. *Business Process Management*. 11(6). Emerald: 695-708.
20. Shishkov B., Xie Z., Liu K., Dietz J.L.G. 2002. Using Norm Analysis to Derive Use Cases from Business Processes. *Proceedings of the 5th Workshop on Organizational Semiotics (OS 2002)*. Delft, The Netherlands. 187-195.
21. Soffer P. 2004. On the Notion of Flexibility in Business Processes. *Proceedings of the 5th BPMDS Workshop*, Porto, Portugal.
22. van der Aalst W.M.P. 2001. How to Handle Dynamic Change and Capture Management Information: An Approach Based on Generic Workflow Models. *Computer Systems, Science, and Engineering. 16(5):295-318*.
23. Vickers Sir G. 1987. *Policymaking, Communication, and Social Learning*. Transaction Books. New Brunswick, NJ.
24. Weinberg G.M., Weinberg D. 1988. *General Principles of Systems Design.* Dorset House. New York.