

## **In Search for a Good Theory: Commuting between Research and Practice in Business Process Domain**

Ilia Bider<sup>1,2</sup>

<sup>1</sup> IbisSoft ab Sweden, [ilia@ibissoft.se](mailto:ilia@ibissoft.se) <sup>2</sup>DSV, Stockholm University, Sweden,

**Abstract.** Kurt Levin's statement “There is nothing more practical than a good theory” says not so much about what is good for practice as what it means to have a good theory. There exist a number of competing theories in the business process domain. The current paper is devoted to one of them that lies outside the mainstream direction. The purpose of the paper is not to present the theory as such, but to present the stages of how it was developed with the aim to become a “good” theory from the practical point of view. The paper is written as an experience report and goes through different stages of the development where research efforts were intermixed with practical tests. The theory in question is the state-oriented view on business processes. The basic idea of this theory lies in application of the general principles from the theory of dynamic systems to the business domain. The main direction for practice in the theoretical development is creating IT-support for loosely structured business processes. Besides giving the history of the related research and practical efforts, the paper discusses the lessons learned that can be of interest for the development of other theoretical models in the business process domain.

**Keywords:** business process, theory, practice, dynamic system

### **1 Introduction**

“There is nothing more practical than a good theory” wrote Kurt Levin [1], p. 169. Two questions arise in connection to this statement. Firstly, how do we know that the theory is good? Secondly, how to create a “good theory”? As far as the first question is concerned, we can refer to the story from the Nobel lecture of the famous physicist R. Feynman [2]:

“... One day a dispute arose at a Physical Society meeting as to the correctness of a calculation by Slotnick of the interaction of an electron with a neutron using pseudo scalar theory with pseudo vector coupling and also, pseudo scalar theory with pseudo scalar coupling. He had found that the answers were not the same, in fact, by one theory, the result was divergent, although convergent with the other. Some people believed that the two theories must give the same answer for the problem. This was a welcome opportunity to test my guesses as to whether I really did understand what these two couplings were. So, I went home, and during the evening I worked out the electron neutron scattering for the pseudo scalar and pseudo vector coupling, saw they

were not equal and subtracted them, and worked out the difference in detail. The next day at the meeting, I saw Slotnick and said, "Slotnick, I worked it out last night, I wanted to see if I got the same answers you do. I got a different answer for each coupling - but, I would like to check in detail with you because I want to make sure of my methods." And, he said, "what do you mean you worked it out last night, it took me six months!" And, when we compared the answers he looked at mine and he asked, "what is that  $Q$  in there, that variable  $Q$ ?" (I had expressions like  $(\tan^{-1}Q)/Q$  etc.). I said, "that's the momentum transferred by the electron, the electron deflected by different angles." "Oh", he said, "no, I only have the limiting value as  $Q$  approaches zero; the forward scattering." Well, it was easy enough to just substitute  $Q$  equals zero in my form and I then got the same answers as he did. But, it took him six months to do the case of zero momentum transfer, whereas, during one evening I had done the finite and arbitrary momentum transfer. That was a thrilling moment for me, like receiving the Nobel Prize, because that convinced me, at last, I did have some kind of method and technique and understood how to do something that other people did not know how to do. ..."

The last sentence in the citation above explains exactly what a good theory is. In the light of this explanation the original citation from K. Levin can be interpreted not only as a statement about what is good for practice, but also as a definition of what a good theory is, i.e. a theory that can be useful in practice.

The second question (how to work out a good theory) is not easy to answer, there can be many possible options. Without pretending to give an answer, the paper presents an experience report on searching for a "good" theory in the business process domain by "commuting" between research and practice. The experience in question concerns the development of a state-oriented approach for controlling business processes. It stretches over a period of more than 25 years, and includes contribution from many people with whom the author cooperated in practice or in research or in both. The search has not been finished yet, but on the way we have worked out and tested in practice a number of hypothesis, some of which still hold and others do not. The author believes that the experience may be useful for others in two respects:

- as an example of "commuting" between research and practice
- which hypothesis have been tested and which of them held and which of them fell (the latter might be more interested than the former)

The paper is structured according to the periods where research or practice were predominant. It starts with the initial theory development (Section 2), goes to the practical test (Section 3) based on which a specialized theory was developed (Section 4) and tested in practice (Section 5).

When the last test in practice showed that we went into the wall, instead of going back to revise the theory, we removed self-imposed restrictions of the theory, and continue experimenting in practice based on the intuition. After the experiments showed that the new direction holds, we went back to theory and revised it according to the experience. After the revision, we went back and implemented the result in practice. This latest period of the theory revision is discussed in Section 6. The concluding Section 7 is devoted to general discussion on interconnection between research and practice and lessons learned.

## 2. Developing the initial theory

The event that triggered our long search was as follows. In the beginning of 1980<sup>th</sup>, the author was responsible for supply of scientific information for the research laboratory he was working for at the time. My duty, in this respect, was to go to the technical library once a week, browse new magazines, and make copies of articles that might be of use for the research and practical work conducted inside the laboratory. Sometime in 1984, I came across the survey of Meyrowitz and van Dam [3] on interactive editing. The paper, actually, was more than just a survey as it promoted the idea that an editor should not be a passive program, but an active system helping its users in their work, e.g., by suggesting continuation as in the syntax-driven editors.

After reading the survey myself, I gave it to my friend and colleague Maxim Khomyakov, who became so excited that he arranged a series of discussions on the topic. These discussions, in the end, resulted in creating an informal project to develop a theoretical framework for designing of what we called “human-assisted systems”. The idea was explained with the two pictures, one representing traditional at the time human-assisting systems (see Fig. 1 to the left), the other one a new paradigm of “human-assisted systems (see Fig. 1 to the right).



Figure 1. Human-assisting vs Human-assisted systems

In a *human-assisting system* (see Fig.1 to the left), a computer helps a human being to perform certain tasks, e.g. to write a letter, to print an invoice, to complete a transaction, etc. The relations between these tasks, and the aim of the whole process are beyond the understanding of the computer, but are a prerogative of the human participant. In a *human-assisted system* (see the Fig.1 to the right), the roles are somewhat reversed, the computer has some knowledge about the process and keeps it running as long as it can. When the system cannot perform a task on its own or figure out what to do next, it will ask the human participant for assistance. The human-assisted system frees human beings from tedious, routing work, like searching for information, bookkeeping, reporting, allowing them to concentrate on thing at which they are best, i.e. decision making.

The main idea of human-assisted systems was that the users and the system should work in a symbiosis. The symbiosis should be flexible which means such cooperation between the system and its users where the distribution of responsibilities between them may change in time. It means that the points of interaction between the system and its users may change, thus we need to have a model in which such changes do not

require substantial modifications. The latter requires a model in which both human and system actions are represented uniformly on equal footing.

The project was of theoretical nature. We had one example in mind though, creating a computer programmer's secretary, a system that would help a programmer to managed his/her job, i.e. to ensure that the programmer does not forget to compile and test after making changes in the source code. The “secretary” should considerably extend the capability of the tools like make/build that existed at the time. The project continued for two years from 1984 to 1986 with three main participants, Maxim Khomyakov (initiator), Eugene Pushchinsky, and the author. The result was a model that consisted of the following components:

- a set of atoms,
- a set of objects
- a code of laws and
- a set of connectors, each connector hanging on a group of objects that must obey a certain law.

Objects have complex structure expressed by including in their “bodies” a set of connectors that hang on other objects making the latter sub-objects to the former. An object's body can also include a connector hanging on the object itself. The dynamics of the objects-connectors model can be defined by a machine in which a connector is regarded as a processing unit that monitors its operands. A connector.

- *awakes* when one of its operands has been changed,
- *checks* whether the law still holds by reading the condition,
- *restores* it when it has been broken,
- *falls asleep*.

A law can be fully deterministic, or not. Non-determinism can concern the condition of awakening, or rules of restoring the low, or both. A connector with a non-deterministic law is called a boundary connector. A boundary connector cannot do its job alone a need help, and here is where human being are introduced in the model. Humans are parts of boundary connectors to help when to awake, or/and how to restore the state of the objects entrusted to this connector.

As we mentioned above, a connector can both be included in the body of an object, and “hang” on this object. This allows an object to reconfigure itself based on changes in other objects. Such reconfiguration can include adding new connectors or removing the existing ones, including the one that completes the reconfiguration itself.

The model itself was published later as a theoretical platform [4,5], and partly [6] (appendixes b and C). The model itself has never been implemented in practice (so far), but served as guidelines for developing specialized theoretical and practical approaches in the domain of business processes.

### **3. Testing the initial theory in practice**

We could not find an occasion to apply the object-connector model to a practical task until 1989, when the author started to work for a small Swedish IT-consulting

company. The company had developed a system to assist salespersons in the pharmaceutical industry, and my task was to support and further develop this system.

A salesperson in the industry was driving around the country, meeting doctors at various hospitals and leaving them prove samples of new medicines. The system was intended to keep order in his/her business, e.g. plan the trips, have a track which samples were left in which hospital, and when it is time to follow up the previously made contact. It also helped to gather statistic and analyze sales potential. The business, hence the system, were build around such concepts as, activity, plan activity, report execution of activity, plan follow-up actions, like phone call.

While working with the system, I started to make a model of the sales business in the pharmaceutical domain in terms of object-connectors model. In this model, a sales lead on which a sales-person is working is represented as an object. As sub-objects, it included a hospital, and a doctor to whom this particular lead is related. A planned activity is represented by a boundary connector that wakes up at the deadline point and ask the salesperson to complete it. While executing the activity, the sales person writes a report and plan farther activities. In terms of the model, the boundary connector that represent the planned activity removes itself from the body of the object placing at the same time some other connectors (new planned activities) instead of it. Part of this activities could be calculated based on the rules, others are set manually by the sales person behind the “steering wheel” of boundary connector.

The DealDriver was initially meant for a company selling goods via telephone. The system was to support both selling via phone calls, and following up customers, and taking and processing orders via phone. The latter included delivery, invoicing, reminders, and payment registry. The selling process was a lighter version of the above description. For order processing we design a special object that represented the current state of processing. This object was presented to the end-users in the form of Fig. 2.

The screen in Fig. 1 represents how much of the order has been processed so far, what has been ordered, how much of ordered has been delivered, whether some money has been invoiced and or payed. This screen represents a so-called static part of the order object. This is complemented by a dynamic part represented in Fig. 2. To the end-user this part is represented as a plan of activities that correspond to the state of the order processing. In our model it is a set of boundary connectors fired by their deadlines and completing some work with the human assistance. Part of this work includes removing itself from the plan, and adding some new activities instead based on the emerging state of the static part of the order object. These new activities could be planned automatically, or added manually by the end-user. Dealdriver included hard-corded automatic planning of next steps, e.g. from order, to packing and delivery from delivery to invoicing (or rest delivery), from invoicing to registering payment (or sending a reminder), etc.



time. For example, numeric dimensions that can be used for the state space for the order process from Fig. 2 can be defined as follows:

- In the first place there are a number of pairs of product-related dimensions  $\langle \text{ordered}, \text{delivered} \rangle$ , 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.
- In addition, there are two numeric dimensions concerning payment: invoiced (amount of money invoiced) and paid (amount of money already received from the customer).

Each process instance of the given type has a goal that can be defined as a set of conditions that have to be fulfilled before a process instance can be considered as finished (end of the process instance trajectory in the space state). A state that satisfies these conditions is called a *final state* of the process. The set of final states for the process in Fig. 2 can be defined as follows: (a) for each ordered item  $Ordered = Delivered$ ; (b)  $To\ pay = Total + Freight + Tax$ ; (c)  $Invoiced = To\ pay$ ; (d)  $Paid = Invoiced$ . These conditions define a “surface” in the state space of this process type.

The process instance is driven forward through *activities* executed either automatically or with a human assistance. Activities can be planned first and executed later. A *planned activity* records such information as type of action (goods shipment, compiling a program, sending a letter), planned date and time, deadline, name of a person responsible for an action, etc.

All activities planned and executed in the frame of the process should be aimed diminishes the *distance* between the current position in the state space and the *nearest* final state. The meaning of the term distance depends on the business process in question. Here, we use this term informally. For example, activities to plan for the process in Fig.2 can be defined in the following manner:

- If for some item  $Ordered > Delivered$ , *shipment* should be performed, or
- If  $To\ pay > Invoiced$ , an *invoice* should be sent, etc.

All activities currently planned for a process instance make up the process *plan*. The plan together with the current position in the state space constitute a so-called generalized state of the process, the plan being an “active” part of it (engine). The process plan on Fig. 3 corresponds to the process instance state shown on Fig. 2. The plan plays the same role as the derivatives in the formalisms used for modeling and controlling physical processes in the Mathematical systems theory. Plan activities shows the direction (type of action) and speed of movement (deadlines), exactly what the first derivatives does in the continues state space.

With regards to the generalized state, the notion of a *valid* state can be defined in addition to the notion of *final state*. To be valid, the generalized state should include all activities required for moving the process to the next state towards the goal. A business process type can be defined as a set of valid generalized states. A business process instance is considered as belonging to this type if for any given moment of time its generalized state belongs to this set. This definition can be converted into an

operational procedure called *rules of planning*. The rules specify what activities could/should be added to/deleted from an invalid generalized state to make it valid. Using these rules, the process instance is driven forward in the following manner. First, an activity from the plan is executed and the position of the process instance in the state space is changed. Then, the plan is corrected to make the generalized state valid; as a result, some actions may be added to the plan and some be removed from it.

The idea of the state oriented view on business processes was first presented in 2000 [11]. More on this view see in [12],[13],[14].

## 5. Back to practice

The state-oriented view on business processes defined in the previous section was tested in practice for two tasks:

- Business process modeling
- Development of business process support systems

As far as business process modeling is concerned, we created a method of modeling processes that an alternative to drawing activities/workflow diagrams. The main steps in this methods are as follows:

- Design a state space for a given business process using mockup screens of the kind of Fig.2
- Listing activities to be completed in the process. Each activity has informal definition on what state of the process it should/could be planned or executed, what should be done (what state will emerge after the execution), who should be doing it (roles)
- Creating examples of the process instances run. For this end we created a special tool Process Visualizer. Using this tool, the user could create a trace of a process instance trajectory by inputting values in a mockup screens. Besides the state screen, we used a plan screen (as in Fig. 3), and an event screen (completed activity). After building such a trajectory, one could follow it step by step in forward or backwards order seeing how planned activities are converted in the state changes and new activities planned.

We have completed over 10 of such process modeling projects in the public sector and non-for-profit organizations. We mostly targeted so called loosely-structured projects, like decision-making, recruiting, negotiations, investigations, etc. Some of the results from these project were published as case studies and experience reports, see [15],[16], and [6, chapter 4]. Our experience showed that the state-oriented approach was quite suitable for modeling loosely structured process and the resulting model could be understood and evaluated by business experts who participated in these processes.

For most of projects (but not all of them) the goal was twofold: (a) to differentiate and understand the process, and (b) create requirements for a business process support

system. For some of the processes we analyzed these requirements were implemented in our new system called ProBis developed for a Swedish interest organization in 2003-2006 as described in [17]. ProBis continued the architectural ideas of DearDriver underpinned by the state-oriented view on business processes and a rich graphical environment (DealDriver was developed for character-based displays). Much more attention was paid for creating a convenient environment for manual planning. ProBis included support for a number of different processes, but used a standardized way of presenting the generalized state of the current instance to the end-users.

The generalized state of the process is presented to the end-user as a window divided in several areas by using the tab dialogues technique, see Fig. 4. Some areas of the window are standard, i.e. independent from the type of the business process; others are specific for each process type supported by the system. Standard areas comprise such attributes and links as:

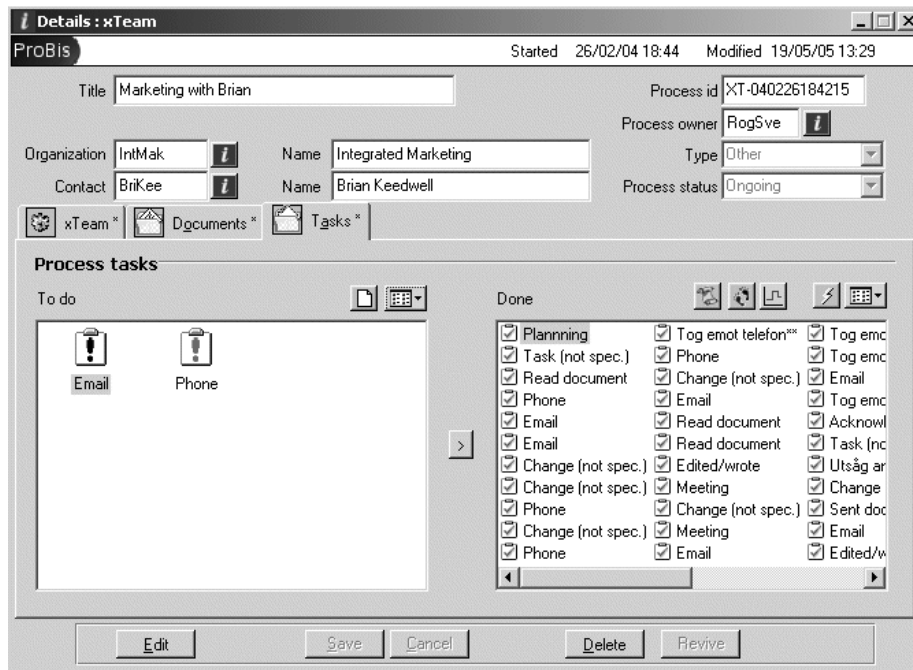
- Name and informal description of a process instance
- Links to the owner, and, possibly, the process team
- Links to the relevant documents, created inside the organization, and received from the outside

The standard part of the generalized state presentation in ProBis includes also the task area (tab) that contains two lists, as in Fig. 4. The *to-do* list (to the left on Fig. 4) includes tasks (synonym for activities from Sections 3,4) planned for the given process instance; the *done* list (to the right on Fig.4) includes tasks completed in the frame of it. A planned task defines what and when something should be done in the frame of the process instance, as well as who should do it. All tasks planned for a given person from all process instances are shown in the end-user's personal calendar. From the calendar, the user can go to any shared space for which he has a task assigned to him/her in order to inspect, change, or execute this task.

Tasks iPB are used as a way of communication between process participants. He communication is done by assigning a task to the another user via filling a special task form. One chooses the task from the list, assigns it to another user of the system, adds a textual description, and some parameters, for example, attaches a document that is already registered in the process instance space. The task list is configurable and can be adjusted for each installation and process type.

To further facilitate communication, several more advanced features were added to ProBis. For example, there is a possibility to plan the same task to many users. Each user gets its own task in the calendar and will need to go and complete it independently of other users. Multi-user planning gives a possibility to easily raise attention of several people to some event that has happened in the process instance. Another advanced feature is the "Returned receipt" check-box which ensures that the planner gets a special "Attention" task planned for him as soon as the task he/she has assigned to somebody else has been completed. More details on ProBis see in [17,18,13,14].

Our experience with ProBis that this kind of systems is quite suitable for loosely structured processes when used by a professional team that knows how to use the system quite well. Introduction of such system into operational practice especially with many occasional users is a challenge (see [17,18]).



**Figure 4.** View on the generalized state in ProBis

We found two main drawbacks with ProBis when using it for more structured process or/processes that involve occasional users:

- The dynamic aspect of business processes is poorly visualized. One needs to go through the done-list and browse through the history to get an understanding of how a given process instance is developing in time.
- To use the system puts some requirements on the user, as he/she needs to understand the general ideas built in the system and get some training. This means that the system is not very friendly for newcomers and casual users. Planning as a way of issuing invitations causes the major problem here, as it is considered to be counter-intuitive. Detailed planning is not as widespread in business life as one can imagine.

We found that these two drawbacks above considerably hamper the possibility of utilization of systems like ProBis for structured processes with many occasional or untrained users. This is especially true in the current business environment where end-users more or less refuse to read manuals and demand the system being so intuitive that one can fully understand it by using try-and-error techniques. This observation has led us to rethinking the whole concept and designing a new way of structuring shared spaces that is better suited to the purposes mentioned above.

## 6. New revision

A new period in our practical and theoretical development started when we moved from the Windows environment to WEB in connection to development of a new business process support system for the municipality of Jonköping. The system was aimed at supporting investigations on suspected child abuse. Instead of porting ProBis to a new technical platform, we started from scratch based on our practical experience. Several decision were made in the beginning of the new project:

- Instead of developing a process support system we decided to create a tool that allows the non-technical people to define their own processes and automatically get WEB-based support for them
- We dropped the idea of using low-level activities (tasks) planning as a primary mechanism for driving the process forward
- We abandon the idea of using tab dialogs for structuring the state space. Instead, we accepted the idea of grouping state parameter (dimensions) in blocks based on the order in which their value are to be identified

After initial test in practice we discovered that the end-users liked the system and considered the blocks we identified as a kind of a process map. Based on the positive feedback, we revised our theoretical ideas that concerned the tool and implemented them in the tool. These work has not been finished yet and we continue to revise both theory and practice.

The tool got the name iPB (interactive Process applications Builder) [18,19,20]. Its purpose is to serve as a platform for building support for loosely structured business processes. The support for a particular process type is achieved by creating this process definition in the hart of which the is a process map. A process map in iPB is a drawing that consists of boxes placed in some order, see Each box represents a step of the process, and the name of the step appears inside the box (no lines or connectors between the boxes). A textual description is attached to each step that explains the work to be done. Each process instance gets its own copy of the map that serves as a table of contents for its shared space, see Fig. 5.

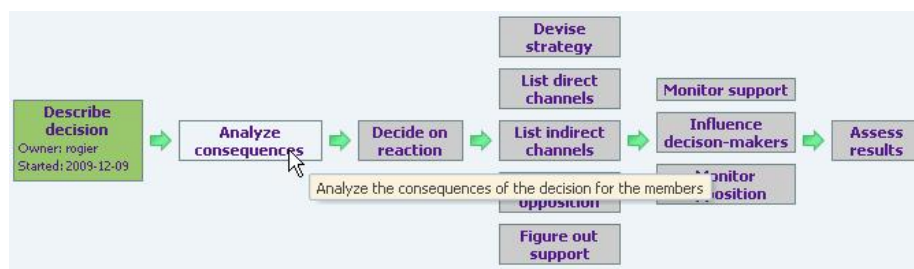


Figure 5. A map used for structuring structuring the state space in iPB

The map is used for multiple purposes: as an overview of the case, guidelines for handling the case, and a menu for navigating inside the shared space. The user navigates through the shared space by clicking on the boxes of the steps with which he/she wants to work. Not all boxes are clickable at the beginning; those that are

grayed require that one or several previous steps are dealt with first, see Fig. 5. These constraints are defined with the help of so-called business rules. A click on a step box redirects the end-user to a web-form that assists him/her in completing the step. The form contains text fields, option menus and radio-buttons to make choices, checkboxes, as well as more complex fields. The form may also include “static” texts that explain what should be done before one can fill some fields.

The progress in filling the step forms is reflected in the map attached to the shared space via steps coloring. A gray box means that the step form has not been filled and cannot be filled for the moment. A white box means that the step form is empty but can be filled. A step with a half-filled form gets the green color, and additional information about when the work on it has been started, and who started it. A step with a fully filled form gets the blue color, and additional information about the finish date.

From the theoretical point of view, the approach described above represents a modification of our state-oriented view on business processes. The basic ideas behind this modification consist of the following:

- The total process state-space is divided into a number of subspaces called process steps. The steps are graphically represented to the end-users as boxes. Subspaces may or may not intersect. The structure of a step subspace is represented to the end-users as a form to fill. Intersecting subspaces means that web forms attached to different steps may contain the same field(s). Usually, in this case, the intersecting fields can be changed only in one form; they are made read-only in the second one.
- The steps are ordered in a two-dimensional matrix that defines a recommended strategy of movement in the state space. The movement starts in the top leftmost subspace and continues in the top down left to right order. This matrix does not prohibit any other way of movement through the subspaces. For example, it allows parallel movements in several subspaces. The matrix is presented to the end-users in the form of a process map.
- The restrictions on movement through the subspaces are defined with the help of business rules. Such a rule, for example, may require that movement in one subspace should be finished before the movement in another one can be started. Business rules are represented to the end users via gray boxes – steps that cannot be handled yet. Clicking on a gray box results in a message that explains why the box is gray, e.g. that some other box should be started first.

## **7. Discussion and lessons learned**

In the previous sections, we described a journey aimed at developing a “good” theory in the business process domain in the sense it was define in Introduction. The journey has not been finished yet, and whether it was successful or not depends on the subjective judgment. However, this is an experience that we can analyzed and draw positive as well as negative lessons from it.

During all stages of the journey, we had some practical aims/examples on which various hypothesis where tested. Even on the initial theoretical stage we had a

“programmer secretary” in mind when debating different options of building a theoretical model. Without this condition, the journey described could not have achieved any progress.

Having a highly abstract theoretical framework in the back of our minds gave us a possibility to see things in reality differently from what they were on the surface. Without having an object-connector model from Section 5, it would be impossible for us to analyze the sales business activity in the way that lead to the development of the state-oriented view on business processes. Such high-level abstract models as the objects-connectors one is difficult to apply directly as they are, but they serve as a perfect guidelines to see the world in an unusual perspective.

Our state-oriented view on business processes was influenced by the Mathematical systems theory that investigate processes in the physical world. Transferring the ideas from one domain to another quite often lead to good theories and is recommended by the general systems theory [21]. However direct “borrowing” a theory from one domain into another will hardly work. In our case we borrowed only the ideas of movement in a state space, without trying to transfer the apparatus of differential equation or state-transition machines. “Borrowing” should go on the conceptual level, the details should be designed taking into consideration the nature of the new domain.

A straightforward application of the theory to practice might not always work in the business process domain. While using planning for driving the process forward was quite natural for people working in sales, it was considered as counter-intuitive for the processes participants in other domains. While making people to accept a completely new way of seeing things is possible, this creates a considerable hinder that is best to avoid.

Our experience with iPB shows that its way to present and support business processes is considered quite natural by many people. In our opinion, two iPB properties contribute to this. Firstly it uses highly visualized and very simple process map to represents the overall position in the state space to all people participating in the process. Secondly, it uses the idea of forms to represent the details. A form is a familiar concept for most modern societies and thus can be easily understood by, practically, everybody. The lesson learned here is that applying a theory in practice where people, not machines, are the main driving force requires presenting new ideas in a familiar to them form. The latter gives better chances for success of the new theory in practice.

At the end of the ProBis period, we found that introducing this kind of system is (even when possible) creates certain difficulties on the flow. Instead of trying to find a theoretical solution, we loosen our attachment to theory and started free experimenting in practice. After this experimenting gave some positive results we went back to adjust the theory. Two lessons can be drawn from this experience. Firstly, narrow following the theory may lead to the dead-en. Secondly, free experiment can give an idea how to develop the theory.

Our experience also shows that a proper theory will stands, even when the facts are against it. To make the theory compatible to the fact, one may need to abandon some of its narrow “axioms” [21]. In our case it was the idea that driving of the process instances through what we call the detailed dynamic distributed planning is mandatory for applying state-oriented view on business processes to practice. Freeing

ourselves from this “axiom” and concentrating on the interplay of subspaces in a way “saved” the theory.

Our theoretical and practical thinking was influenced by a number of ideas and systems that we found in other domains, e.g. mathematical system theory, programming language REFAL, the general systems theory to name a few. None of them, however was taken from the business process domain. Not following the mainstream in the domain, helped us to be outside the workflow box and see alternative solutions for what we call loosely structured business processes for which the mainstream does not have any good suggestion. The lesson learned here is that following too much what is happening in one's own domain may be harmful for creativity. Looking beyond it into the adjustment domains may give a better clue how to continue in one's own domain.

The above does not mean that we do not know, or reject everything that has been and is being done in the business process domain. In [22] we classified different views on business processes, and pointed out the areas of applicability for each of them. We believe that each view has its own area of application and should be developed and tested according to it, and we hope that our general lessons listed above could be useful even when developing this view.

In conclusion we want to name a more general principle that we followed on our journey. There are two ways of scientific investigation of reality:

- With minimum interference, i.e. through observation and measurement
- With maximum interference, i.e. by applying a considerable force and seeing how the object under investigation behaves.

While both methods are legitimate and widely used, there is a practical limit of what can be achieved through using only the first one. If we deal with a complex system, its structure and behavior may not reveal itself unless under a stress. This is why we have chosen the second method combining research and practice in the setting of a consulting company whose customers were from time to time interested and willing to try some new ideas in their organizational practice.

### **Acknowledgments**

This paper would have never been written without considerable efforts of the team of researchers, developers, business-experts and end-users with whom the author worked in research and practice. Not having the space to name all of the participants of the long journey, I especially wish to express my gratitude to my nearest colleagues Tomas Andersson, Alexander Durnovo, Paul Johannesson, Maxim Khomyakov, Erik Perjons, Eugene Pushchinsky, Alexey Striy, Rogier Svensson.

### **References**

- [1] K. Lewin, Field theory in social science: Selected theoretical papers by Kurt Lewin. London: Tavistock. 1952 (p.169)
- [2] R.P. Feynman. Nobel Lecture, December 11, 1965  
[http://nobelprize.org/nobel\\_prizes/physics/laureates/1965/feynman-lecture.html](http://nobelprize.org/nobel_prizes/physics/laureates/1965/feynman-lecture.html)

- [3] N. Meyrowitz and A. van Dam. Interactive Editing Systems. *ACM Computing Surveys*, 14(3), 1982, 321-415.
- [4] I. Bider, M. Khomyakov, and E. Pushchinsky, Logic of change: Semantics of object systems with active relations, *Automated Software Engineering (ASE)*, vol. 7, no. 1, 2000, pp. 9-37.
- [5] I. Bider and M. Khomyakov, New technology - Great Opportunities. How to Exploit Them. *Enterprise information systems IV*, J. Filipe, Ed. Kluwer, eds., 2003, pp. 11-20.
- [6] I. Bider, *State-oriented business process modeling: principles, theory and practice*. PhD thesis, KTH (Royal Institute of Technology), Stockholm, 2002.
- [7] I. Bider, Developing tool support for process oriented management, *Handbook of Systems Development*, vol. 1999, CRC Press, 1998, pp. 205-222.
- [8] I. Bider, Object driver: a method for analysis, design, and implementation of interactive applications, *Handbook of Systems Development*, vol. 1999, CRC Press, 1998, pp. 81-96.
- [9] I. Bider and M. Khomyakov, Object-oriented model for representing software production processes, *ECCOOP'97 Workshop Reader*, Springer, LNCS 1357, 1997, pp. 319-322
- [10] R.E. Kalman, P.L Falb, and M.A. Arbib, *Topics in Mathematical System Theory*. McGraw-Hill, 1969.
- [11] M. Khomyakov and I. Bider, Achieving Workflow Flexibility through Taming the Chaos, in OOIS 2000-6th international conference on object oriented information systems, Springer, 2000, pp.85-92.
- [12] B. Andersson, I. Bider, P. Johannesson and E. Perjons, Towards a Formal Definition of Goal-Oriented Business Process Patterns. *BPMJ*, Emerald, V11(6), 2005, pp. 650 - 662 .
- [13] G. Regev, I. Bider, A. Wegmann. Defining Business Process Flexibility with the help of Invariants. *SPIP*, Wiley, V12(1), 2007, pp 65-79.
- [14] I. Bider and A. Striy, "Controlling business process instance flexibility via rules of planning," *IJBPM*, vol. 3, no. 1., 2008, pp. 15-25
- [15] T. Andersson, A. Andersson-Ceder and I. Bider, State Flow as a Way of Analyzing Business Processes – Case Studies. *Logistics Information Management*, Emerald, Vol.15 (1), , 2002, pp. 34-45
- [16] E. Perjons, I. Bider, B. Andersson, Building and Exploiting a Business Process Model for Lobbying: Experience Report. *Communications of the IIMA (CIIMA)*, Vol. 7. No 3., 2007 , pp. 1-14.
- [17] T. Andersson, I. Bider, and R. Svensson, Aligning people to business processes experience report, *Software Process Improvement and Practice (SPIP)*, vol. 10, no. 4, 2005, pp. 403-413.
- [18] I. Bider, E. Perjons, and P. Johannesson, In Search of the Holy Grail: Integrating social software with BPM. Experience Report, *Enterprise, Business-Process and Information Systems Modeling*, LNBIP #50, Springer, 2010, pp. 1-13.
- [19] I. Bider, E. Perjons, and P. Johannesson, A strategy for integrating social software with business process support , 12 p. (to be published in LNBIP #66, Springer, 2011).
- [20] iPB Reference Manual on-line documentation, <http://docs.ibissoft.se/node/3>. [Online]. [Accessed: 20-Jun-2010].
- [21] G.M. Weinberg, *An Introduction to General Systems Thinking*, Dorset House, 2001
- [22] I. Bider and E. Perjons, Evaluating Adequacy of Business Process Modeling Approaches. *Handbook of Research on Complex Dynamic Process Management: Techniques for Adaptability in Turbulent Environments*, IGI 2009, pp. 79-102