Conceptual Process Modeling Language: Regulative Approach

Oleg Svatoš
Department of Information Technologies, University of Economics, W Churchill sqr. 4, 130 67 Prague 3, Czech Republic,

Keywords: analysis and design, process modeling languages, conceptual modeling, business process modeling.

Abstract:
In my research I try to introduce little bit different perspective incorporated into Process Modeling (PM) than contemporary approaches provide. I try to look at processes through tool used by people for centuries - rules and regulations. From this perspective I challenge contemporary technical approaches by an approach that, from my perspective, better captures the substance of a business process as relations among people and not machines. I suggest a new type of modeling notation that incorporates this perspective and also incorporates other features that I miss in contemporary approaches.

1 Introduction
Process modeling and especially Business Process Modeling is a very active filed these days. The businesses are trying to find a way how to keep their control over their dynamically expanding organizations and IT departments are trying to help them to achieve this goal. It is no wonder that many Information System development methods and techniques are spreading into managerial levels. There are two main directions process modeling is heading. One represented by the Business Process Reengineering (BPR) that draws the IS design methods up to the managerial levels and second represented by IS design and especially Service Oriented Architecture (SOA) and service integration (web-services).

Contemporary methodologies and methods of BPR are well presented in [1] including overview of BPM (Business Process Modeling) techniques/languages. The goal is to capture the business processes, analyze them and introduce changes that make processes more effective, measurable and kept under control. Overview specifically focused on business process modeling languages with a good background can be found in and [4]. There are many sources on modeling languages itself, but there are few on their comparison. For one of the few good comparisons of conceptual languages see [5].

The SOA approach [13] to business process modeling is different. The goal is to withdraw the business logic out of the code and keep it in different easily manageable/changeable form – models that can be seen as configuration files for web services orchestration or choreography. Well described in [3]. If there is not only service to service cooperation, but also people are involved, models are called workflows [2]. These models extend the process executing engines by human interface so that the IS users are part of a process.

Interesting thing is that both these branches use the same modeling languages. The question is whether IS modeling tools are appropriate for managerial high-level task or they introduce more problems one has to deal with.
2 Purpose and Objective of the Research

There are two types of users that use process modeling. In this paper I will differentiate them as process analysts and process designers. These terms became nowadays so vague, they have to be defines first.

- The goal of the process analyst is to identify and capture the process that is already being performed in the reality with or without knowledge of its existence.
- The goal of process designer is to take already identified and captured processes and make them formal – ready for optimization and automatization.

Here we have two groups of users with different interests, way of analysis, way of working with reality. The most visible difference in meanings we can find in the idea of what a process as such is.

Process designers see the process as “A collection of related, structured activities--a chain of events--that produce a specific service or product for a particular customer or customers.” The important word here is structured. That is what designers do. They take relatively well structured process, turn it in some formal model that can be later on used for automatization through some workflow or process execution engine. Modeling language for such use requires well founded (logically, mathematically) semantics, that can be converted into machine language.

Unfortunately the practice had taught us that in reality there are very few well structured processes. Growing complexity of business puts us in different position. In this complex world, we have to first understand the part of the reality we are analyzing, than identify the core elements and model an abstraction in relevant semantics, knowing the limits of the language we use. That is what the process analysts do. The model they produce is a conceptual model and it is a base for structuring down the captured process.

Guizzardi [12] cites John Mylopoulos who defines the discipline of conceptual modeling as

"the activity of formally describing some aspects of the physical and social world around us for purposes of understanding and communication. Conceptual modeling supports structuring and inferential facilities that are psychologically grounded. After all, the descriptions that arise from conceptual modeling activities are intended to be used by humans, not machines... The adequacy of a conceptual modelling notation rests on its contribution to the construction of models of reality that promote a common understanding of that reality among their human users."\(^4\)

Understanding the conceptual process modeling this way brings several problems into our consideration.

The goal of the research is to discuss appropriateness of contemporary process modeling languages, commonly used for formal description of processes in BPR, for high-level (conceptual) modeling and introduction of a new process modeling language for conceptual process modeling.

3 Contemporary Process Modeling Languages

The research recognizes following process modeling languages: BPMN [6], Activity Diagrams [7], BPEL[9], XPDL [10], EPC [14], IDEF3 [16], PSL [11], YAWL [15].

---

2 WfMC
3 BPEL
4 Critique of Contemporary Process Modeling Languages

4.1 Focus on Capturing Reality
The first problem of contemporary modeling languages is their convertibility into machine language. This feature is not for free. In different words, they are derived from machine language and modeling a process in them means programming a process. It is fine when thinking how to create a BPEL code for execution, but it is a disaster when trying to understand the reality and model a part of it. When creating conceptual model one should focus on the right understanding of the reality and not to spend the time by thinking, how to capture the reality in the modeling language used. The use of the conceptual language should be natural and independent on later usage (implementation).

4.2 Function vs Activity
Another core problem is the vision of the world. The computer world sees every change as a product of a function. No matter what are these changes called (transitions, activities, tasks) they are still seen as functions. The only differentiation we can find here is simple (atomic) function and complex function. Unsurprisingly a complex function is consisting from atomic functions. The idea is that there are atomic functions and the process of modeling is to disassemble the complex functions (activities) into the atomic ones. Again this is maybe fine with the design but not with the analysis.

When analyzing reality, looking for atomic functions means getting somewhere at subatomic level. That is not lucky idea to get that low. Atomic functions are applicable only in computer programming. In conceptual modeling one has to bear in mind that there are only activities and all are complex. They are no functions.

4.3 Function and Activity States
From programmers point, function can be executed, it can finish ok or produce an error or an exception. It is simple, it is obvious, but one can find it only in very few modeling languages – BPEL and BPMN (BPDM). Yes, functions have states and these states are caused by events. In case of functions all events, except the one starting the execution, are of internal origin. We do not know what has caused an error, exception or why has the function finished successfully. In case of functions it is irrelevant to capture the causes since it is depended on execution environment and other factors beyond our control.

From process analyst point (or at least from my point), activity can be started, finished, stopped, continued, cancelled or failed. Since at conceptual level all the activities can be considered complex, it makes sense to let the activity states to be caused by external events. There is no execution environment – we are trying to capture a part of reality and we should be interested what events cause activity state changes.

4.4 Plans and Duties
Looking at process model, one can see a plan. A plan, who should do what and when. These plans also contain different scenarios that are either used or not, depending on whether a specific event occurs or a specific state of affairs has become true.

This brings an interesting question up. Is a plan consisting only from duties (that is what activities are) are not there any rights for involved parties?

5 Regulative Approach to Conceptual Process Modeling
In this chapter I will outline the conceptual modeling language that tries to implement into modeling critique done in previous chapter. This outline only deals with the event/activity flow modeling. The business object, inputs, outputs, organization units, etc. are not subject to this outline.

Requested Features Summary: conceptual language is focused on reality analysis, understandable and usable by non-programmers, all activities are complex, activities occur in time and there are states an
activity can be in, there are events that cause transition to specified activity state, there are duties and rights and there are complex decisions.

Following the requests, the notation could look like in Figure 1.

![Figure 1: Basic Concepts of Regulative Approach](image)

An Event is either related to a duty, right or a fact. If we group events by their content, we can get a Grouped Events element that is very similar to the element Activity, we know from other modeling languages.

Every event has a mark that specifies what change of state of duty or right or fact (reality) the event signifies. The six recognized states of an activity allow expressing more than regular approaches. All states are caused by external events - we specify events that change activity states. This is in contrast with states caused by internal events - where an error or an exception is caused by an unknown force behind the function execution (BPMN). The “fact” event is in a special position here. Model cannot change the reality. In that sense there is no reason to specify events that cause “fact” events. On the other hand its declarative nature may be useful in some other cases – judge’s decision whether someone has or has not done some activity, etc.

The state sign in a “fact” event represents:

- The **Start** and **Finish** state events allowing us to better specify when the transition should come – right after the activity begins or when it is completed. Other modeling languages are either ambiguous (do not capture this detail) or there has to be regular language used to draw the difference (past tense, adding the term start into activity description, etc.). **Stop** state event specifies that the activity is interrupted and **Continue** that the activity is continued. **Cancel** state event specifies that the activity has been permanently stopped. **Fail** state event specifies that the activity has finished unsuccessfully.

For the “Duty” and “Right” events is the meaning bit different since the states refer to either desired or claimable states of affairs.

We will consider, same as in BPMN, three types of events (Figure 2): Start, Intermediate and End. This differentiation is necessary to make clear, from the process analyst point of view, where the process starts, what events influence it and where the process ends.

![Figure 2: Three Types of Events](image)

The Figure 3 shows the model in basic form. Events causing other events to rise. If an event is a fact, type of event is left blank. For analytical use this is diagram too complicated since it is too hard to understand what these “groups of events” mean in the bigger picture. Now we try to group state events of the same activities. The diagram (figure 4) looks far clearer than the figure 3. We have events and activities here same as in other modeling languages. However, activities here are in form of rights and duties. Grouping of events has not lost any detail, since differentiation among start, intermediate and end events is preserved.
Although, the model may look similar to regular ones, there are many differences. First of all it captures not only duties but also rights. Customer’s right to cancel the order is fully incorporated into the process. There is specified when this right starts, when and why it finishes or when and why it is canceled. This all is specified in context of the part of “Making Order Process” that is pictured here. Same as rights, the duties have specified what starts them and when and why they are considered finished. For instance the duty Verify the Order. It is clearly specified what starts and especially ends this “activity” without necessity to model this “activity” in greater detail.

6 Conclusions

In this work I have tried to show how contemporary process modeling languages are inappropriate for real conceptual process modeling. Their sensitivity on easiness of structurability of modeled reality makes them hard to use for capturing the reality. They push the analyst into thinking how he is going to model a process instead what he is going to model.

Following my critique I suggest a different approach, “Regulative Approach”, to conceptual process modeling based on thousands of years used technique – contracts. This approach recognizes that actors do not have only duties (activities) but also rights and that at cognition level all activities are complex. Activities are treated as entities that go through different states, which are caused/changed by events originating inside or outside of the modeled process. I suggest six possible activity states: Start, Finish, Stop, Continue, Cancel and Fail. Considering these six states of an activity, occurrence of activities in this approach is expressed by events that signify the change of an activity state. The activity flow itself then consists from chains of events as the states of activities are changed by explicit events over time.
This modeling language tries to be easily human-readable and understandable (events causing other events) without technology related concepts. It tries to save analyst’s time and let one focus on what is essential. For specifying essential events analyst doesn’t have to diagram a sub-process since all activities (Grouped Events) are treated as complex. Differentiation of duties and rights makes the diagram richer and more complete. It focuses on what is the purpose of the modeled system – Duties and rights define more of particular goals than list of actions what to do (depending on the level of detail, of course). Actors can be assigned rights and duties and this allows proper role permission definition through process models. This way the modeling language allows capturing how these permissions change dynamically as the actual process proceeds including worse case scenarios – what happens if an actor fails to complete his duty.

Regulative approach presented here is just at its start point. There is lot to improve and enhance. This paper should give the idea what makes it different from contemporary modeling process languages and why it has sense to work out this concept into more sophisticated method.

References: