Modeling Complex Time Limits

Oleg Svatos

Department of Information Technologies,

University of Economics, Prague, Czech Republic

svatoso@vse.cz

Abstract: *In this paper we analyze complexity of time limits we can find especially in regulated processes of public administration. First we review the most popular process modeling languages. There is defined an example scenario based on the current Czech legislature which is then captured in discussed process modeling languages. Analysis shows that the contemporary process modeling languages support capturing of the time limit only partially. This causes troubles to analysts and unnecessary complexity of the models. Upon unsatisfying results of the contemporary process modeling languages we analyze the complexity of the time limits in greater detail and outline lifecycles of a time limit using the multiple dynamic generalizations pattern. As an alternative to the popular process modeling languages there is presented PSD process modeling language, which supports the defined lifecycles of a time limit natively and therefore allows keeping the models simple and easy to understand.*

Keywords: time limit, business process modeling, process state diagram, EPC, BPMN, multiple dynamic generalizations pattern

1. Introduction

Time limits are undividable part of almost any business process. The main reason is that most of the business processes are about delivering their outputs (Davenport, 1993, p. 5) to their customers in certain time since it is crucial to deliver the output not only in the specified quality but also in time. Whereas in commercial business processes the usage of the time limits is usually fairly simple (time limits usually just start and finish at in advance specified times), there are special areas of business process modeling like civil proceedings and administrative proceedings, which procedures are heavily regulated by many regulations and where the time limits play significant role. This is then reflected in their complexity. The time limits are not only being started and finished but also extended, suspended, etc. Correct capturing and correct understanding of the captured time limits within a business process is in this area crucial for correct management of the business process in order to be able to get the intended output and to avoid procedural or even material problems caused by failing some time limit due to ambiguity in the business process model.

2. Time Limits in Business Process Modeling Languages

In the three most popular contemporary process modeling languages (Becker, Breuker, Weiß, & Winkelmann, 2010), which we will review here, the time limits are either explicitly supported (UML, BPMN) or they allow implicit support through “individual implementation” by other available concepts (eEPC).

2.1 BPMN

A time limit is represented in BPMN (Object Management Group, 2011a) by the timer event symbol (Figure 1) which is also used to capture a time point or a repeating time moment. The time event is thrown and then caught by the timer events in case the time limit expires or the specified time point occurs. No other time limit event is in the BPMN associated with the timer.

Differentiation of the time point and time limit in a model is dependent on associated text with the timer symbol.

The time limit length is set usually by the text associated with the timer symbol. If the time limit length is variable there can be used the data object symbol (Figure 2) to hold its value.

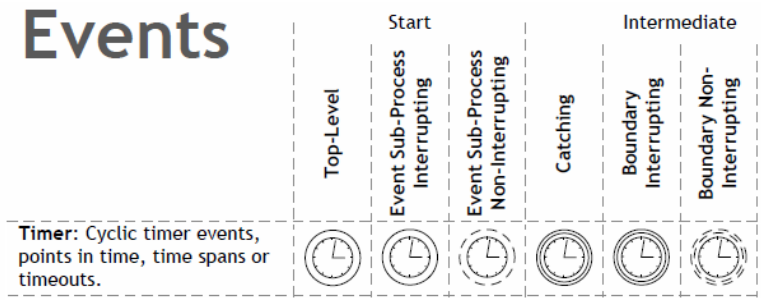

Figure 1: BPMN Timer Events (Berliner BPM-Offensive, 2010)

Figure 2: BPMN Data Object

2.2 UML Activity Diagram

Time limit is represented in the activity diagram (Object Management Group, 2011b) by the accept time event action symbol (Figure 3). This can represent (same as in the BPMN) not only a time limit but also a time point or a repeating time moment. The time event is caught by the accept time event action in case the time limit expires or the specified time point occurs. No other time limit event is in the activity diagram associated with the accept time event action.

Figure 3: UML Activity Diagram Accept Time Event Action

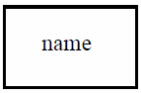

Figure 4: UML Activity Diagram Object Node

Differentiation of the time point and time limit in a model is dependent on associated text with the accept time event action symbol.

The time limit length is set usually by the text associated with the accept time event action symbol. If the time limit length is variable there can be used the object node symbol (Figure 4) to hold its value.

2.3 eEPC

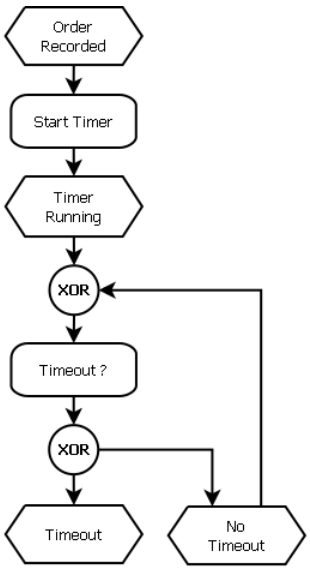

Figure 5: eEPC Time Limit Implementation (Davis & Brabänder, 2007)

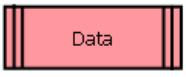

Figure 6: eEPC Data Input/ Output

eEPC (IDS Scheer AG, 2010) does not have any specific symbol for the time limit or time point. Since there is no time limit concept in the eEPC, it has to be either implemented by a combination of general events and functions (Figure 5), for instance as in (Davis & Brabänder, 2007), or the general event symbol can be used to represent timer expiry event similarly to the BPMN or UML activity diagram. Differentiation of the time point and time limit in a model is dependent on associated text with the event or function symbol.

The time limit length definition is dependent on the time limit implementation. It can be set by the text associated with the function or event or, if the time limit length is variable, there can be used the data symbol () to hold its value.

3. Example Scenario

In order to see how the contemporary process modeling languages can deal with capturing time limits in a business process we will define and use a scenario based on Act No. 137 of 14 March 2006 on Public Contracts (Czech Republic, 2006) which describes the process of making a public contract. We will simplify the real process for the purposes of this analysis and we will use only reasonable part of this process (Figure 7) with focus on two time limits: time limit for submission and award period.

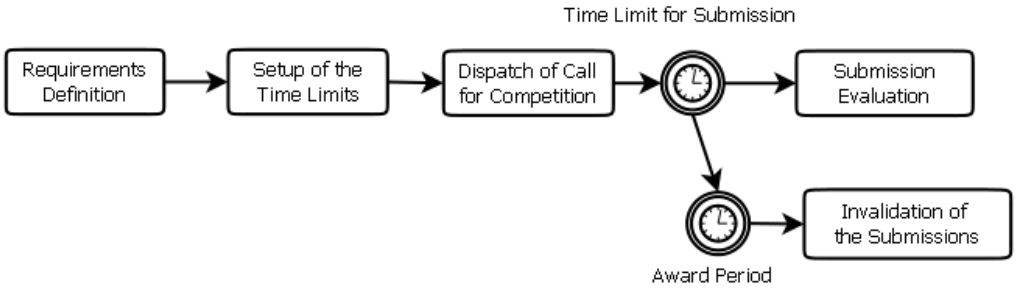

Figure 7: Example Process Scheme

The modeled process starts when the requirements for the offers are finished and there can be setup the time limit for submission and the award period. After that there is dispatched call for competition and the time limit for submission starts running.

The time limit for submission is:

- set by the activity Setup of the Time Limits
- started next day to the day the call for competition was dispatched
- extended when published notification is altered
- reduced if the notice was sent by electronic means

When the time limit for submission expires the award period starts to run.

The award period is:

- set by the activity Setup of the Time Limits
- started when the time limit for submission expires
- suspended when there are raised objections
- continued when the objections are resolved
- interrupted when the Office for Protection of Competition orders corrective measures to be taken[1]
- restarted after the corrective measures had been taken

The process further continues in selection of the best offer or resolving the situation if the offers get non-binding.

The example is divided into two parts in order to discuss the time limit for submission and the award period separately as each part is focused on different aspects of the time limit.

The analysis in the chapter 2 shows that the UML activity diagram is very similar to BPMN in the way of capturing of the time limits and therefore there are no models in UML activity diagram in this chapter

[1] This is modification of the regulation in order to include an interruption in the model. In the regulation is the award period suspended.

since modeling the example in UML activity diagram would have no additional benefits for the analysis. Results of the BPMN models are applicable to UML activity diagram.

3.1 Time Limit for Submission

eEPC

Figure 8 captures the eEPC model of the example part focused on the time limit for submission. This model represents the "event" approach to the time limit implementation.

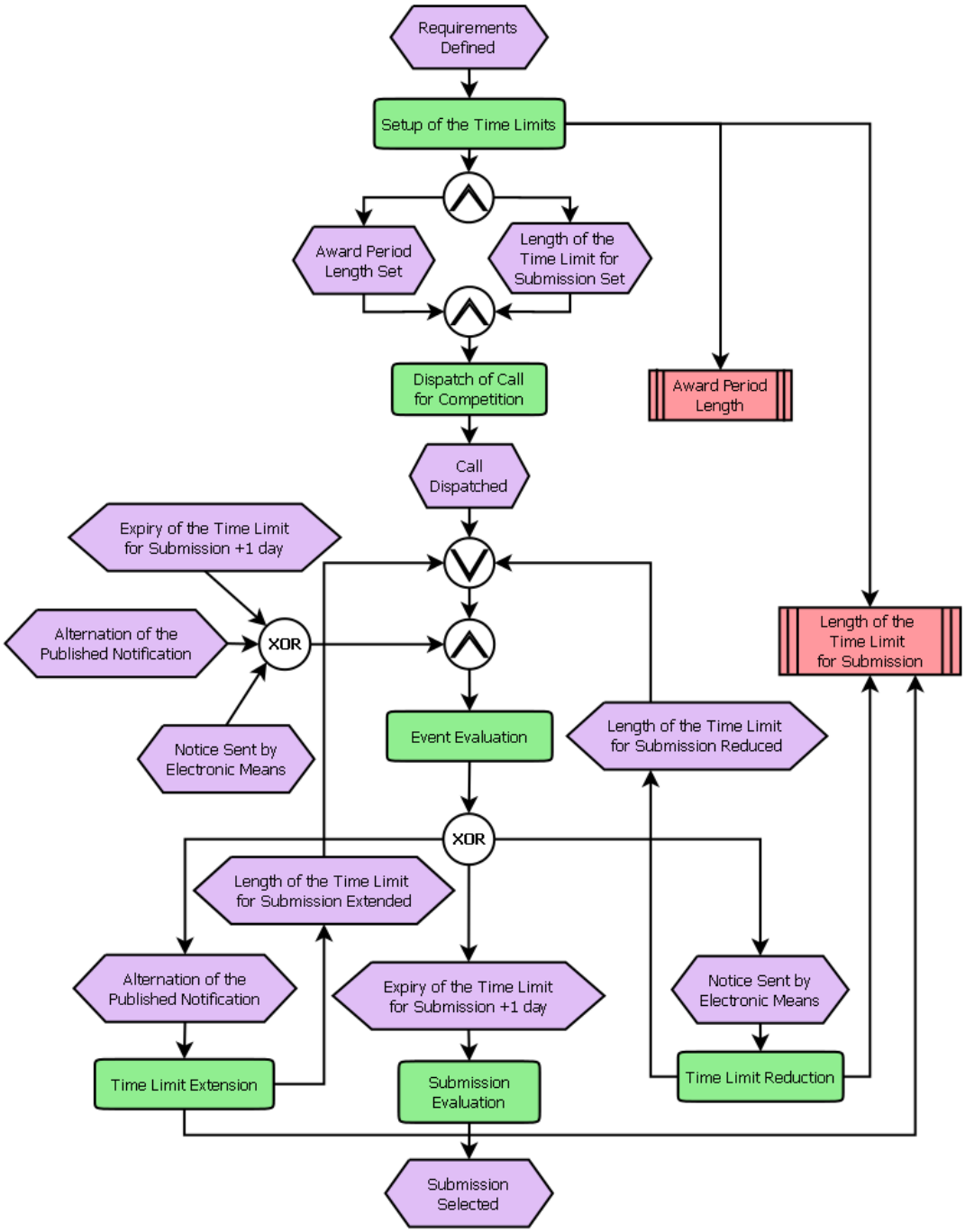

Figure 8: eEPC "Event" Approach to the Time Limit for Submission Model

Following constructions and concerns should be pointed out:

- The event Expiry of the Time limit for Submission incorporates into the model the reaction to the expiration of the time limit for submission, but the eEPC cannot this way specify when the time limit for submission actually starts.

- There is unclear, what is the relation of the Length of the Time Limit for Submission data and the Expiry of the Time Limit for Submission event. They can be linked together only by referencing the same time limit name and there is no way to tell whether a change of the length of the time limit for submission has also effect on already started time limit for submission.
- States of the time limit for submission and the length of the time limit for submission are spread all over the model. This makes the model hard to read.

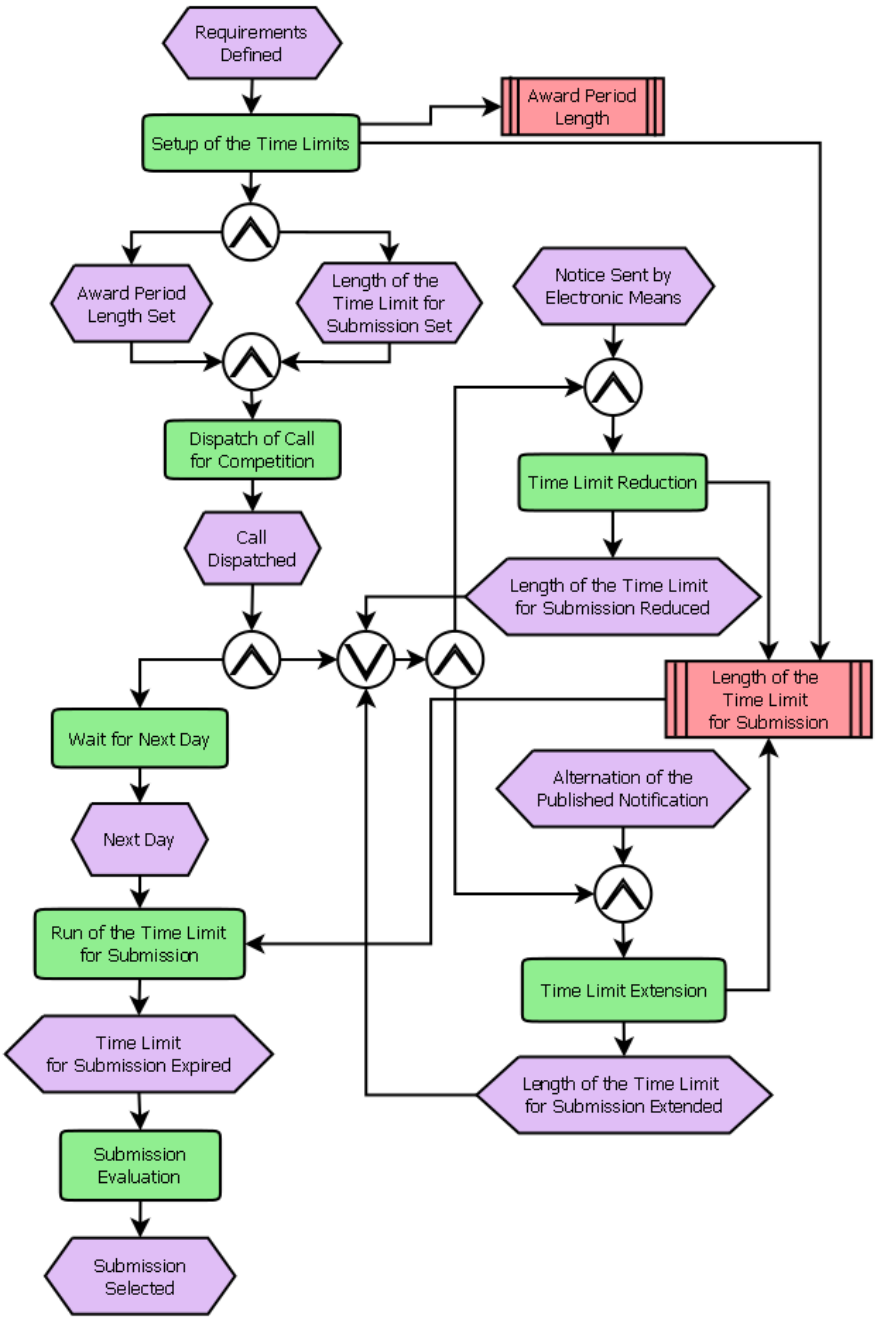

Figure 9: eEPC "Functional" Approach to the Time Limit for Submission Model

Figure 9 captures again the eEPC model of the example part focused on the time limit for submission. This time the model represents the "functional" approach to the time limit implementation. Following constructions and concerns should be pointed out:

- Time limit is implemented as a function which has obvious start and end. This also makes reasonable to capture explicitly waiting of the time limit for submission start for the next day. In previous model this was implicitly included in the time limit itself (+1 day).

- Compared with the previous model (Figure 8), the Figure 9 clearly specifies when the time limit for submission actually starts.
- There is still unclear, what is the relation of the Time Limit for Submission data and the Run of the Time Limit for Submission function. They are linked together by relation and the same name, but there is still no way to tell whether a change of the length of the time limit for submission has also effect on already started time limit for submission.
- Many relations in the model make it hard to read.

BPMN

Figure 10 captures the BPMN model of the example part focused on the time limit for submission. The way of capturing of the time limit length states is inspired by (IBM Corporation, 2006).

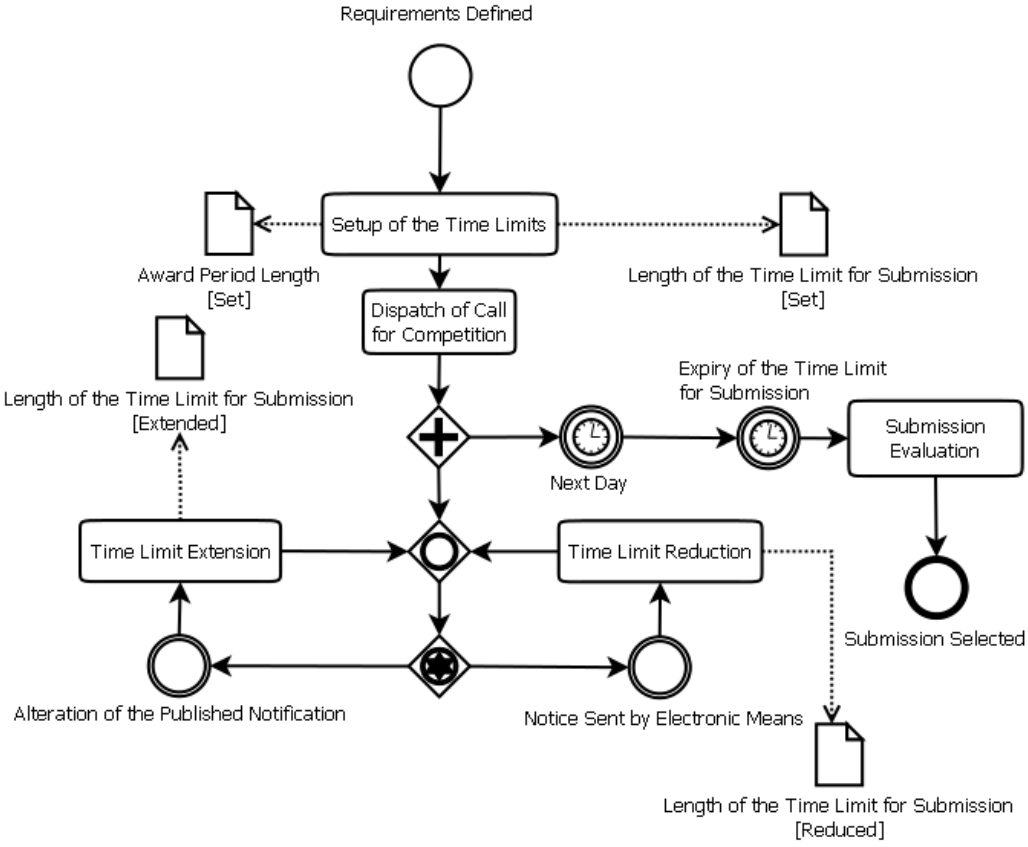

Figure 10: BPMN Time Limit for Submission Model

Following constructions and concerns should be pointed out:

- The BPMN specification does not state clearly when the timer starts. Since the timer is referenced in the specification also as a delay we assume that the timer starts when the incoming flow of the timer catch event symbol becomes active.
- There is unclear, what is the relation of the Time Limit for Submission data object and the Expiry of the Time Limit for Submission timer. They can be linked together only by referencing the same time limit name and there is no way to tell whether a change of the length of the time limit for submission has also effect on already started time limit for submission.

3.2 Award Period

eEPC

Figure 11 captures the eEPC model of the example part focused on the award period. Implementation of the time limit as one function, as in Figure 9, was not possible since suspension and continuation of an activity is in the eEPC very cumbersome, as discussed in (Svatoš, 2012), and a workaround would be in this case unusable. Therefore we had to use the different approach to the “functional” time limit implementation pictured in Figure 5.

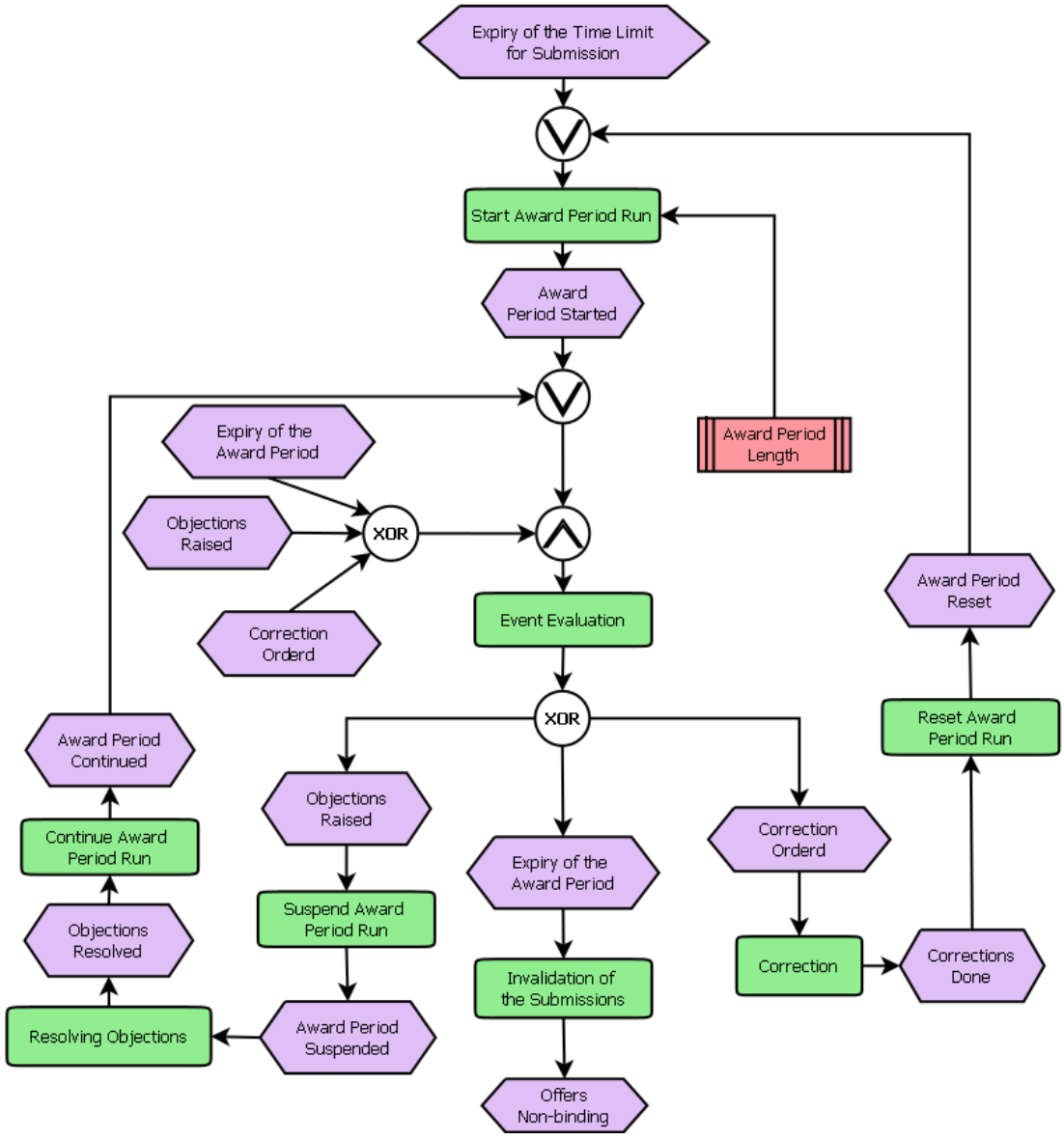

Figure 11: eEPC Award Period Model

Following constructions and concerns should be pointed out:

- States of the award period run are spread all over the model and each state requires having its own function. This makes the model very “talkative” and unclear.
- This type of time limit implementation in the eEPC has all functions and states referencing the award period run (they change its state) but this entity itself is not present in the model. It is only referenced in the text.
- There is unclear, what is the relation of the Award Period data and the Expiry of the Award Period event. They can be linked together only by referencing the same time limit name and there is no way to tell whether a change of the award period length has also effect on already started award period.

BPMN

Figure 12 captures the BPMN model of the example part focused on the award period. The timer concept does not support the idea of suspending and continuing of a time limit. Therefore it has to be extended through the implementation based on the available concepts. This forces us to use complicated and sort of low level implementation of the suspension and continuation of the time limit, since we have to operate with two variables for the award period.

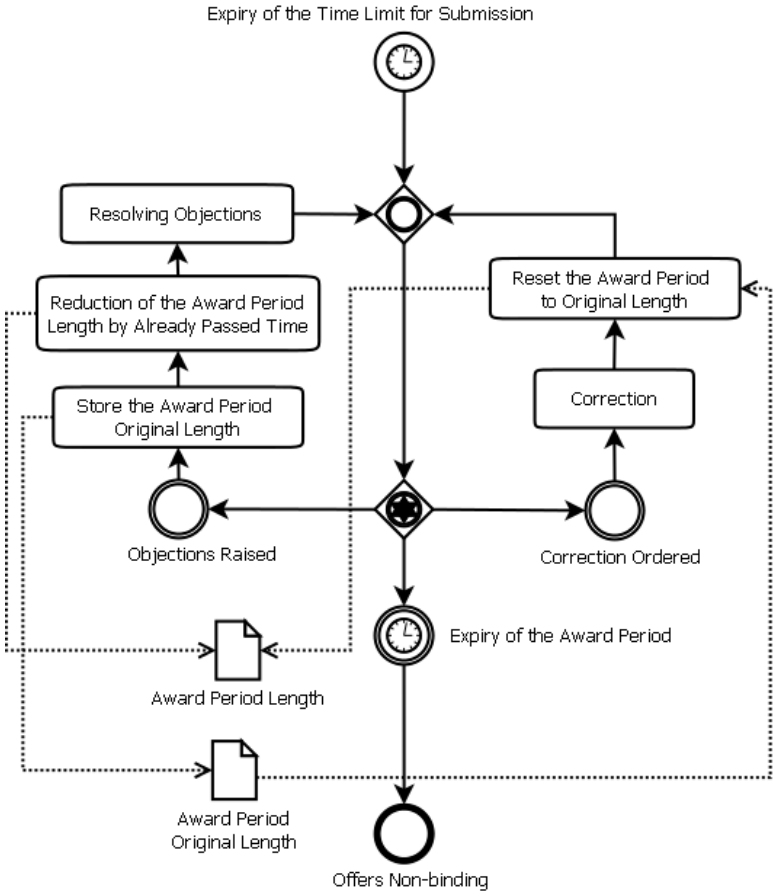

Figure 12: BPMN Award Period Model

Following constructions and concerns should be pointed out:

- Algorithmic model of the suspending and continuing of the time limit adds technical details that have nothing to do with the modeled reality. One has to either guess the real meaning of these constructions or it has to be included as a text comment.
- Dual usage of the time limit data symbol – in case of the time limit for submission the data object is used as a general time limit definition and the activities that are associated with it correspond to the time limit changes in the example definition. In case of the award period scenario, the data object represents length of one actual award period and the activities associated with it are not actual changes of the award period (we cannot find them in the scenario definition). They represent implementation details since in the reality the award period length stays the same. It should not change throughout this scenario, since, what changes, is the state of the award period run.
- There is unclear, what is the relation of the award period data objects and the Expiry of the Award Period timer and which data actually applies to the Expiry of the Award Period timer. They can be linked together only by referencing the same time limit name and there is no way to tell whether a change of the award period length has also effect on already started award period.
- The model would get far more complicated if the extension or reduction of the award period was included.

4. Time Limits and Their Lifecycles According to Law

The models of the example scenario, discussed in the previous chapter, are far from perfect and therefore we have to look at the time limits in greater detail in order to see where the problem is.

The legal theory defines two basic types of a time limit (Ondrýsek, 2010):

- Material time limit (e.g. limitation of time)
- Process time limit (e.g. time limit for appeal)

Material time limit is a time limit in which there has to occur some event with which the regulation links defined material consequences (BusinessInfo.cz, 2009). In other words, the time limit defines the time in which the relevant party has to use its right or it will be time-barred. These time limits are mostly determined by material regulations like Civil Code (Czech Republic, 1964), Commercial Law (Czech Republic, 2012), etc.

Process time limit is a time limit in which the involved parties may or have to perform some action in the ongoing proceedings (BusinessInfo.cz, 2009). These time limits are mostly determined by process regulations like the Code of Civil Procedure (Czech Republic, 1963), the Administrative Procedure Code (Czech Republic, 2004), etc.

Since this paper deals with processes and time limits in them, we will focus further on the process time limits. The goal of the analysis is to find changes related to the time limit which the process modeling languages should be able to capture. As a tool for the analysis we will use the UML state machine diagram (Object Management Group, 2011b) in order to map the lifecycle of a time limit.

When searching for the states that form the time limit lifecycle, we can see that there are two main aspects concerning the time limit that have to be taken into consideration: the time limit length and the actual run of the time limit.

According to the Czech legal theory (Ondrýsek, 2010) and legislature (e.g. Act on Public Contracts (Czech Republic, 2006)) the time limit length can be:

- set
- extended
- reduced

And the time limit run can be:

- started
- interrupted (reset)
- suspended
- passed

Looking at the changes listed above one can see that the changes of the time limit length are independent on the changes of the time limit run and vice versa. These changes therefore constitute two lifecycles. The time limit run lifecycle (Figure 13) and the time limit length lifecycle (Figure 14).

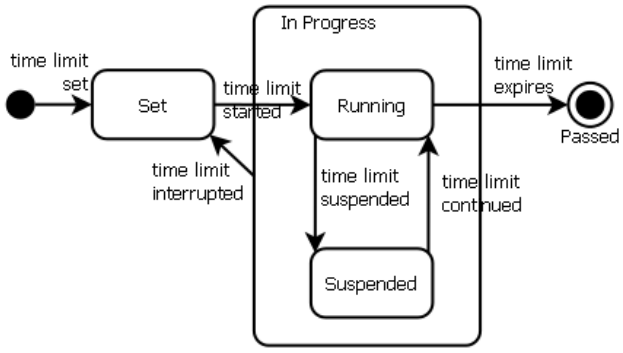

Figure 13: Time Limit Run Lifecycle

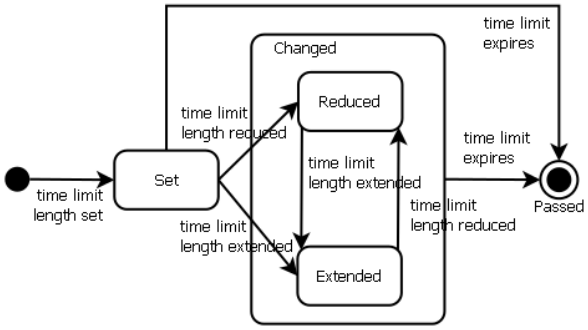

Figure 14: Time Limit Length Lifecycle

They share the *set* state, but these states have different meanings and are part of are two different lifecycles. The *set* state in the time limit length lifecycle represents a fact that there has been formulated time limit length, which can change over time. In the time limit run lifecycle the *set* state represents a fact that the time limit counter has been initialized with some value (constant) or reference (time limit length variable).The time limit length can be set and even changed before it is used to set the time limit counter.

When the time limit run is set, it waits to be started. Immediately as the time limit run is started, the state of the time limit run changes to running. When running, it can be suspended or it may finish. When the time limit run is running or suspended it can be also interrupted and then it would wait to be started again.

The time limit length lifecycle is fairly simple. Right after the time limit length is set it can be either reduced or extended. When the time limit expires the time limit length enters *passed* state.

As noted in (Řepa, 2012), these two lifecycles represent two viewpoints at the same time limit concept. Relation of the individual viewpoints and the original concept can be represented by a class diagram (Figure 15), which is compliant with the rules of the multiple dynamic generalizations pattern.

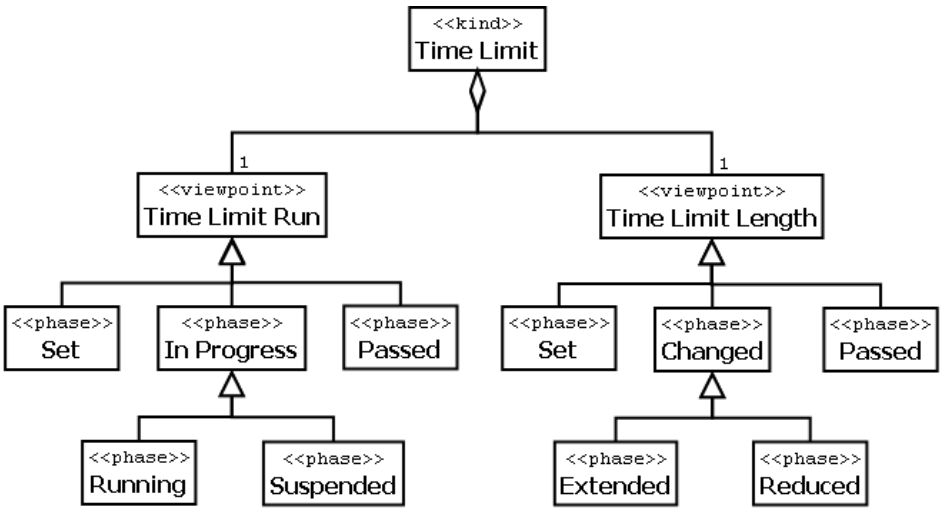

Figure 15: Time Limit Viewpoints

It is the existence of two independent lifecycles of the time limit what makes capturing of the time limits hard in contemporary popular process modeling languages, we have discussed in chapter 3. This is not true for all process modeling languages. There exists an alternative, which respects the complexity of a time limit - the process state diagram.

5. Process State Diagram (PSD)

The process state diagram, proposed in (Svatoš, 2011), provides specific symbol (Figure 17) and defined lifecycles (Figure 16) for capturing time points and time limits. Used lifecycles help also to formally differentiate the time limits and time points in a model.

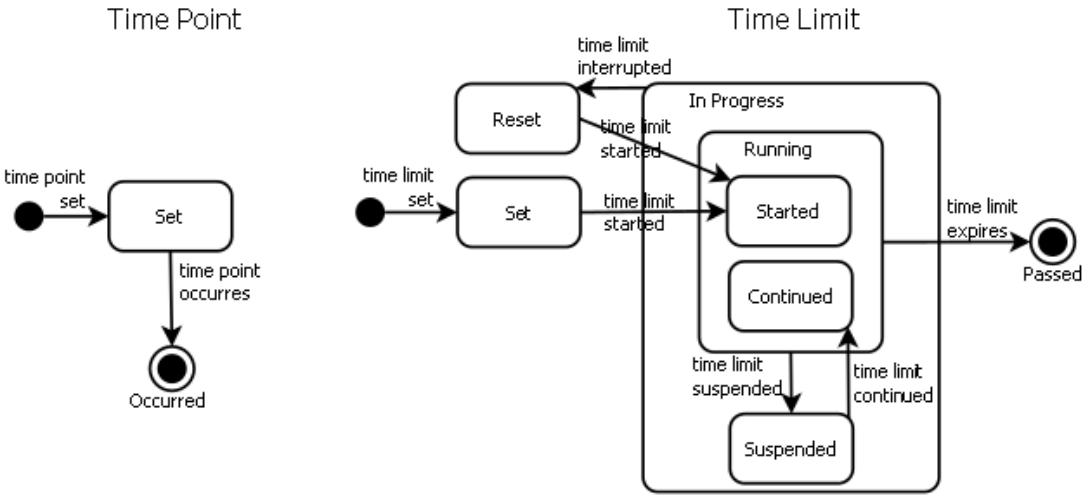

Figure 16: PSD Time Limit and Point States

PSD (Svatoš, 2011, p. 113), in accordance with the analysis done in the chapter 4, distinguishes between the time limit run, which is represented by the time limit/ point symbol (Figure 17), and the time limit length which is represented by the object symbol (Figure 18); each representing the abstract

viewpoint from the Figure 15. If the situation does not require it, the time limit length can be setup by text associated with the time limit/ point symbol.

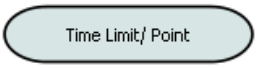

Figure 17: PSD Time Limit or Point Symbol

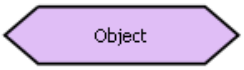

Figure 18: PSD Object Symbol

5.1 Time Limit for Submission

Figure 19 captures the PSD model of the example part focused on the time limit for submission. Following constructions should be pointed out:

- The PSD, thanks to the standardized time limit lifecycles, clearly specifies when the time limit starts and what happens when it expires.
- The set-set relation between the Length of the Time Limit for Submission object and the Time Limit for Submission time limit constitutes in PSD special relation (Svatoš, 2011, p. 113), which specifies the Length of the Time Limit for Submission object as a holder of the actual length value for the Time Limit for Submission time limit meaning that if the value of the length changes it has an effect on already started time limit for submission.
- States related to the time limit for submission run and the length of the time limit for submission are at one place in the model where they should be. They do not “wander” around the model. This makes model clear and easy to read.

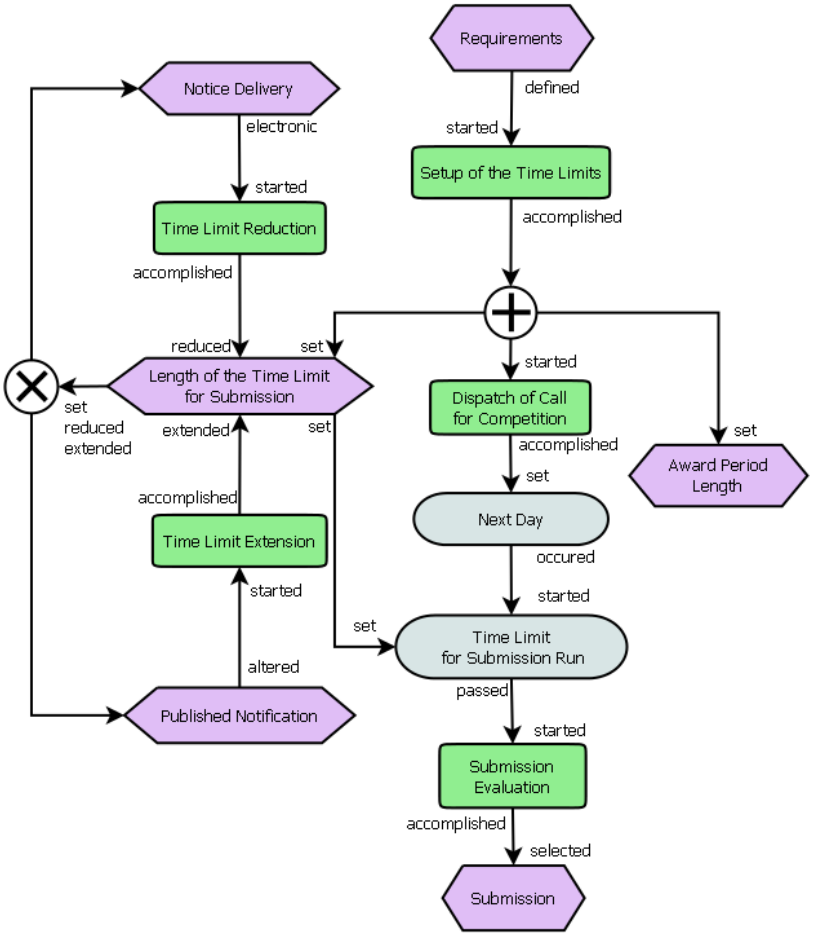

Figure 19: PSD Time Limit for Submission Model

5.2 Award Period

Figure 20 captures the PSD model of the example part focused on the award period. Unlike the predecessors the PSD has full support of the time limit run lifecycle defined in chapter 4. This allows clear capturing of the suspension and continuation of the award period without any workarounds.

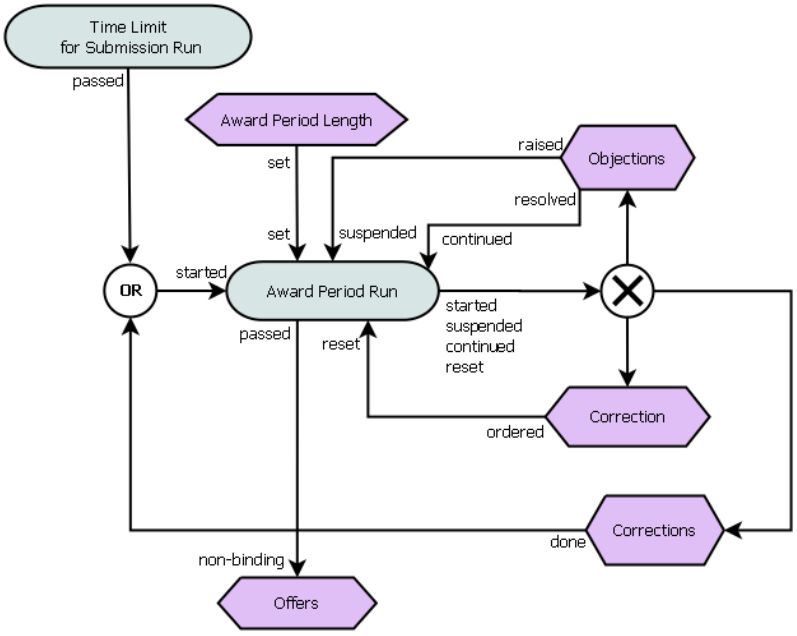

Figure 20: PSD Award Period Model

Following constructions should be pointed out:

- The PSD, thanks to the standardized time limit lifecycles, clearly specifies when the time limit starts and what happens when it expires.
- Award period length definition and actual run are bound by the set-set relation (see the previous chapter). This makes the relation between the award period run and the award period length clear.
- Changes captured in the model are related only to the actual run of the award period. This is consistent with the example definition.
- States related to the award period run and award period length are at one place in the model where they should be. They do not “wander” around the model. This makes model clear and easily understandable.

6. Conclusions

In this paper we focused on capturing time limits in business process models. For this purpose, there was defined an example based on the current Czech legislature which was then captured in popular process modeling languages.

Analysis showed that the popular process modeling languages support the time limits in the defined example only partly. The BPMN, UML activity diagram have a basic concept of a time limit implemented which captures only two time limit states: started and expired. Events like suspension, continuation or a reset of a time limit has to be implemented individually by the analyst. The eEPC has no explicit concept for time limit capturing at all and therefore it is completely reliant on analyst’s decision how one implements changes of the time limit states using the available concepts.

The individual implementation of unsupported time limit states brings unwanted ambiguity and complexity into models including technical details that have nothing to do with the captured reality itself. They just serve the purpose of working around the lack of support and make the models hard to read and understand.

Upon unsatisfactory results of the process modeling languages in the example scenario, we looked at the time limits in greater detail. We analyzed the time limits from the legal point of view, outlined

general lifecycles of a time limit and showed importance of understanding the time limit capturing as a problem that has two parts: the length of the time limit and the actual run of the time limit.

As an alternative to the popular process modeling languages there is presented process state diagram (PSD) and its benefits demonstrated on the example.

In all discussed popular process modeling languages one has to think first about how to capture the time limit before thinking about capturing of the actual time limit properties and yet the properties determine what is the best fitting implementation of unsupported time limit states. This makes the modeling itself unnecessarily complicated. All these difficulties become nonexistent when using the process state diagram. The PSD clearly separates the time limit run and time limit length concepts and fully supports the time limit run and time limit length lifecycles defined in the analysis. This allows a business process analyst to focus only on the modeled problem, keeps time limit capturing in formal form and most importantly keeps the models clear and easy to understand.

References

Becker, J., Breuker, D., Weiß, D., & Winkelmann, A., 2010. Exploring the Status Quo of Business Process Modelling Languages in the Banking Sector – An Empirical Insight into the Usage of Methods in Banks. 21st Australasian Conference on Information Systems (ACIS 2010). Retrieved July 30, 2011, from http://aisel.aisnet.org/acis2010/8/.

Berliner BPM-Offensive, 2010. Retrieved July 22, 2011, from BPMN 2.0 Poster: http://www.bpmb.de/index.php/BPMNPoster

BusinessInfo.cz, 2009. Základy občanského soudního řízení. Retrieved June 2, 2012, from http://www.businessinfo.cz/cz/clanek/orientace-v-pravnich-ukonech/zaklady-obcanskeho-soudniho-rizeni-opu/1000818/49259/

Czech Republic, 1963. Act 99 of December 4, 1963 on Code of Civil Procedure. Retrieved May 30, 2012, from http://www.jafbase.fr/docUE/Slovaquie/COde%20ProcCvi.pdf.

Czech Republic, 1964. Act of the Czech Republic No. 40/1964 Sb. Civil Code. Retrieved May 30, 2012, from http://www.janvalenta.cz/korpus/doku.php?id=civil_code.

Czech Republic, 2004. Act 500 of June 24, 2004 on Administrative Procedure Code. In Collection of Laws of the Czech Republic. Part 174 (pp. 9782-9844. Retrieved July 30, 2011, from http://aplikace.mvcr.cz/sbirka-zakonu/ViewFile.aspx?type=c&id=4478). Prague.

Czech Republic, 2006. Act 137 of March 14, 2006 on Public Contracts. Retrieved May 30, 2012, from http://www.portal-vz.cz/CMSPages/GetFile.aspx?guid=eb0e7fdb-5830-4d90-a3ed-ba5232e82e68.

Czech Republic, 2012. Act 513, 1991 on Commercial Code. Retrieved May 15, 2012, from business.center.cz: Retrieved May 30, 2012, from http://business.center.cz/business/pravo/zakony/obchzak/

Davenport, T. H., 1993. *Process Innovation: Reengineering Work through Information Technology*. Harvard Business School Press. ISBN: 0-87584-366-2.

Davis, R., & Brabänder, E., 2007. *ARIS Design Platform Getting Started with BPM*. London: Springer-Verlag London Limited. ISBN 978-1-8462-8612-4.

IBM Corporation, 2006. October. Introduction to BPMN. Retrieved June 14, 2011, from www.bpmn.org/Documents/OMG_BPMN_Tutorial.pdf

IDS Scheer AG, 2010. ARSI Method: ARIS Platform Version 7.1 – Service Release 8. Retrieved June 12, 2011, from http://documentation.softwareag.com/aris/aris71e/method_manual_aris_s.pdf

Object Management Group, 2011a. Business Process Model and Notation (BPMN) Specification Version 2.0. Retrieved June 12, 2011, from http://www.omg.org/spec/BPMN/2.0

Object Management Group, 2011b. March. UML Version 2.4 - Beta 2. Retrieved September 5, 2011, from Object Management Group: http://www.omg.org/spec/UML/2.4/

Ondrýsek, R., 2010. Čas jako právní skutečnost (Time as a Legal Fact). Právnická fakulta Masarykovy univerzity. Retrieved July 30, 2011, from http://is.muni.cz/th/244973/pravf_m/DP_Cas_jako_pravni_skutecnost.pdf.

Řepa, V., 2012. *Modelling Life Cycles of Generic Object Classes. In Information Systems Development.* New York: Springer.

Svatoš, O., 2011, *Business Process Modeling: Process Events and States*. Praha: VŠE-FIS.

Svatoš, O., 2012. Modeling Suspension and Continuation of a Process. *Journal of Systems Integration* 3 (2). Available at: http://www.si-journal.org. ISSN: 1804-2724.

JEL: K12