Modeling Suspension and Continuation of a Process

Oleg Svatos

Department of Information Technologies,

University of Economics, Prague, Czech Republic

svatoso@vse.cz

Abstract: *This work focuses on difficulties an analyst encounters when modeling suspension and continuation of a process in contemporary process modeling languages. As a basis there is introduced general lifecycle of an activity which is then compared to activity lifecycles supported by individual process modeling languages. The comparison shows that the contemporary process modeling languages cover the defined general lifecycle of an activity only partially. There are picked two popular process modeling languages and there is modeled real example, which reviews how the modeling languages can get along with their lack of native support of suspension and continuation of an activity. Upon the unsatisfying results of the contemporary process modeling languages in the modeled example, there is presented a new process modeling language which, as demonstrated, is capable of capturing suspension and continuation of an activity in much simpler and precise way.*

Keywords: business process modeling, activity lifecycle, process modeling language

1. Introduction

Suspension and continuation of a process is a common event in business processes especially in those that are bound by some regulation and their compliance with the regulation is enforced. In most cases there is a main process which is delivering the intended output and there are several parallel ones that check if the main process is going according the regulation (rules). If not, one of the common approaches, to bring it back to the state compliant with the regulation, is to suspend it, sort things out and then let it continue.

Good examples are processes regulated by the Administrative Procedure Code Act N. 500/2004 Coll. (Act 500, 2004), which defines the general rules and tools for administrative proceedings including conditions when a proceeding has to be suspended and when it can be continued. This regulation has to be applied in all types of proceedings the state administration performs.

For instance it is implemented in the building permit proceedings, which is also regulated by the Building Act N. 183/2006 Coll. (Act 183, 2006) bringing further detail into conditions for suspension and continuation of the proceedings. The building law in addition brings rules and tools for building activity regulation possibility to order suspension of all construction activities until the compliance with regulations is solved. There is specified under which conditions is the suspension optional (office decision) or obligatory.

Similar example is the Civil Procedure Code (Civil Procedure Code, 1963) which defines when should be the court proceedings suspended. In addition the courts themselves have the power to order suspension or continuation of a process/activity.

Suspension can be found in commercial business, too. For example (according to the contract provisions) client can suspend paying for his mortgage (Flexible Mortgage, 2011) for some time or paying to his pension fund (Act No. 42,1994). These are no exceptions, just regular options the client has.

The suspension should not be seen as some internal affair of a process. The suspension has not only effect on the suspended process, but also on all related activities/processes that are effected or even executed upon the suspension or continuation event.

When we talk about suspension and continuation of a process by the same token we can talk about suspension and continuation of an activity since in the process modeling, due to the composite nature of processes (Řepa, 2012), a process is either complex activity with an explicit sub-process, which

may consist from further detailed processes, or an atomic activity with an implicit or non-existent sub-process.

Suspension and continuation of an activity are states an activity may go through its life. There are, of course, more states an activity may go through and these are captured in activity lifecycle diagram.

2. Activity Lifecycle

Activity lifecycle can be captured by a state transition diagram (STD) (Yourdon, 2006) or UML state machine diagram (Object Management Group, 2011a), used in this case. In this work we will discuss only the states and their possible transitions. Analysis of events that can cause the transitions is out of the scope of this work.

Probably the most detailed activity lifecycle analysis we can find in (Weske, 2007). Weske in his work analyzes lifecycle of an activity from business process model designed for execution in execution environment. This process is usually modeled in process modeling language like BPMN (Object Management Group, 2011b) and then transformed into process description in process execution language (for instance BPEL[1]), which is then used for process execution.

Resulting diagram (Figure 2-1) then presents activity lifecycle of a technical activity which depicts the states and possible transitions a technical activity may go through in its life.

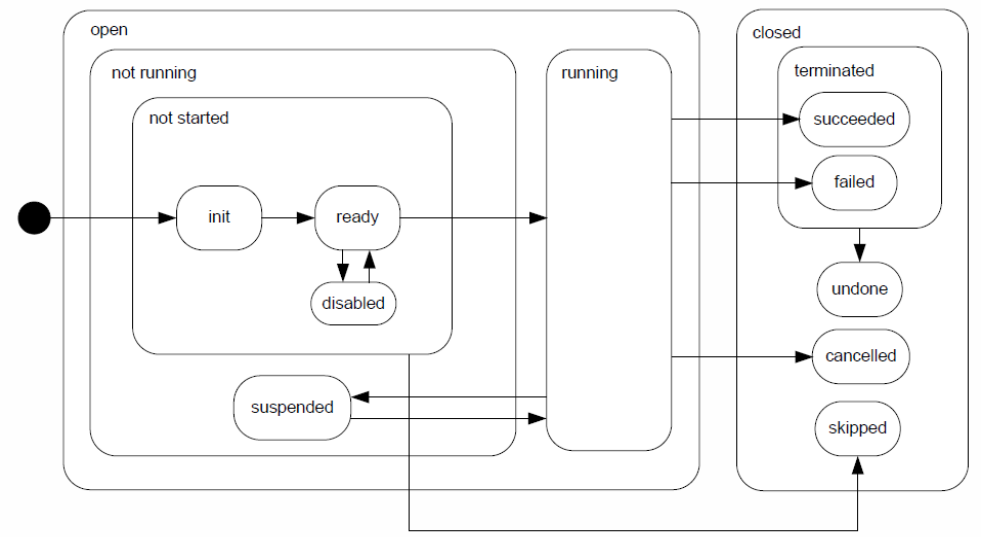

Figure 2-1: Technical Activity Lifecycle *(Weske, 2007)*

Activity in execution environment turns first into *init* state. When it is enabled its state changes to *ready* state. If the activity for some reason is no longer ready to be executed, its state changes to *disabled* and it is returned back to *ready* state when the execution becomes possible. If the activity is not required for further execution it can be *skipped* (In the figure represented as transition from *not started* state, which involves *init*, *ready* and *disabled* states, to *skipped* state).

When *ready*, the activity can be executed and it enters the *running* state. While in *running* state the activity can be *suspended* and later on continued by switching back to the *running* state.

When the activity has finished it enters the *terminated* state. This is either *succeeded* state, which represents accomplishment of the activity's goal or *failed* state, which represents failure of the activity to accomplish the activity's goal. *Terminated* activities can be *undone* using compensation or transaction recovery techniques. Not all activities have to finish, some of them may be *cancelled*. Nevertheless this is also a final *closed* state of an activity.

[1] *Business Process Execution Language* - http://oasis-open.org

2.1 Activity Lifecycle in Contemporary Modeling Languages

As mentioned above the processes for execution are modeled in business process modeling languages. In this chapter we will have a look at how an activity lifecycle is understood by business process modeling languages.

Activity lifecycle diagram as in Figure 2-1 can be outlined for each business process modeling language, describing their point of view at activity lifecycle. We will outline the activity lifecycle diagrams for the two most popular process modeling languages (Becker, 2010).

In EPC (IDS Scheer AG, 2010) are activities called functions. Figure 2-2 describes the lifecycle of EPC's function.

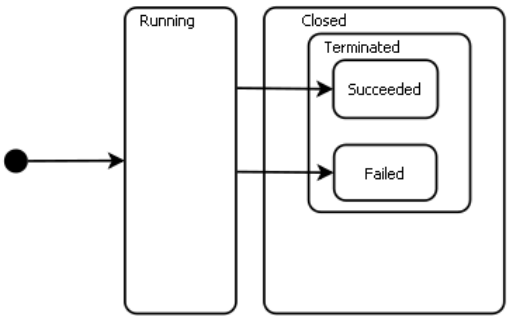

Figure 2-2: EPC Function Lifecycle

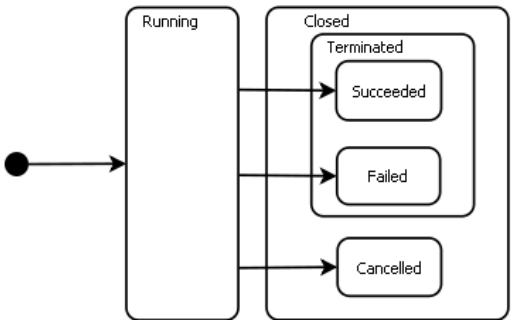

Figure 2-3: BPMN Task Lifecycle

A control flow pointing at a function represents possible start of the function. When the function is started by an activation of the incoming control flow its state changes to *running*. When the function is finished successfully the state of the function changes to *succeeded* and the outgoing control flow, which points to event representing function success, is activated. When the function fails, its state changes to *failed* and the outgoing control flow, which points to an event representing function failure, is activated.

In BPMN (Object Management Group, 2011b) are activities called tasks. Figure 2-3 describes the lifecycle of BPMN's task.

A sequence flow pointing at a task represents possible start of the task. When the task is started by activation of the incoming sequence flow its state changes to *running*. When the task is finished successfully the state of the task changes to *succeeded* and the outgoing regular sequence flow is activated. When the task fails, its state changes to *failed* and the sequence flow outgoing from immediate error event attached to the task border is activated. BPMN can cancel a task directly by attaching an immediate event to the task border. When the defined event occurs, the state of the task changes to *cancelled* state and the outgoing flow from the attached immediate event is activated.

BPMN is half way to having the *undone* state. There is compensation event which may start compensation task which resolves somehow the problem of inconsistency after a task has been interrupted, but the final state, when a compensation activity finishes, is just regular event. There is no special event for finished compensation flow and therefore there is also no undone state in Figure 2-3.

Activity states reasonably differ when comparing the defined technical activity lifecycle (Figure 2-1) and the lifecycle diagram of EPC's activity (Figure 2-2) or BPMN's activity (Figure 2-3). What is missing is the *not started* stage with all its states and the *suspended, undone* and *skipped* states from further stages. Unlike the BPMN the EPC is also missing support of the *cancelled* state[2].

3. Impact Analysis on Real Example

The lack of native support of all activity states does not necessarily mean that the process modeling languages cannot capture these states. It may just require more effort and giving up on accuracy and simplicity. How this can be done and what it requires in case of suspension and continuation of an activity, we will discuss over an example based on real process. This analysis is not biased by the

[2] *This is addressed in EPC extension Yet Another EPC (yEPC) (Mendling, Neumann, Nüttgens, 2005)*

topic of the modeled example since the general problems and ways of capturing of activity states are independent on the sector the process is part of.

Our example focuses on regulation of construction realization activity (the building construction process) based on the Building Act N. 183/2006 Coll. (Act 183, 2006). Conformance of the construction activity with the regulation is evaluated by the building office, which has the power to order a suspension of an activity if the activity violates the regulation.

The building office checks the construction realization activity's compliance with the regulations through on-site inspections at the construction site.

We will recognize four different events that may be the reason for starting scheduling of the on-site inspection.

- First is when the construction realization has started. Whether it is with or without permission may be solved when the inspection takes place.
- Second is when the building permit is issued. The permit also defines in which phase the inspections should take place.
- Third is when the construction realization activity is continued after it was suspended for some time since the conditions for continuation were fulfilled.
- Fourth is when the inspection does not find anything what would require suspension of the construction and so there can be scheduled next inspection.

When the inspection scheduling is finished there is set time when the inspection will take place. The inspection starts at the scheduled time and when finished there can be issued a discontinuation of works order or, if the inspection does not find anything that is in conflict with the regulations, the construction realization activity may go on uninterrupted and the inspection is scheduled again.

If the building office decides after the inspection that there are required necessary construction modifications to be done or differences between the plan and the actual realization need to be approved, the construction realization itself is suspended. This is defined in the Building Act N. 183/2006 Coll. § 134 (Act 183, 2006).

This part of the process is, for the purpose of this example, kept simple. When there is issued discontinuation of works order, the construction realization activity is suspended. There are also defined conditions for continuation in the discontinuation order, which have to be addressed by the process of defects rectification. When they are fulfilled, the construction realization may go on. The continuation of the construction realization activity also means start of scheduling another on-site inspection.

3.1 Model in EPC

Figure 3-1: Defects Rectification in EPC

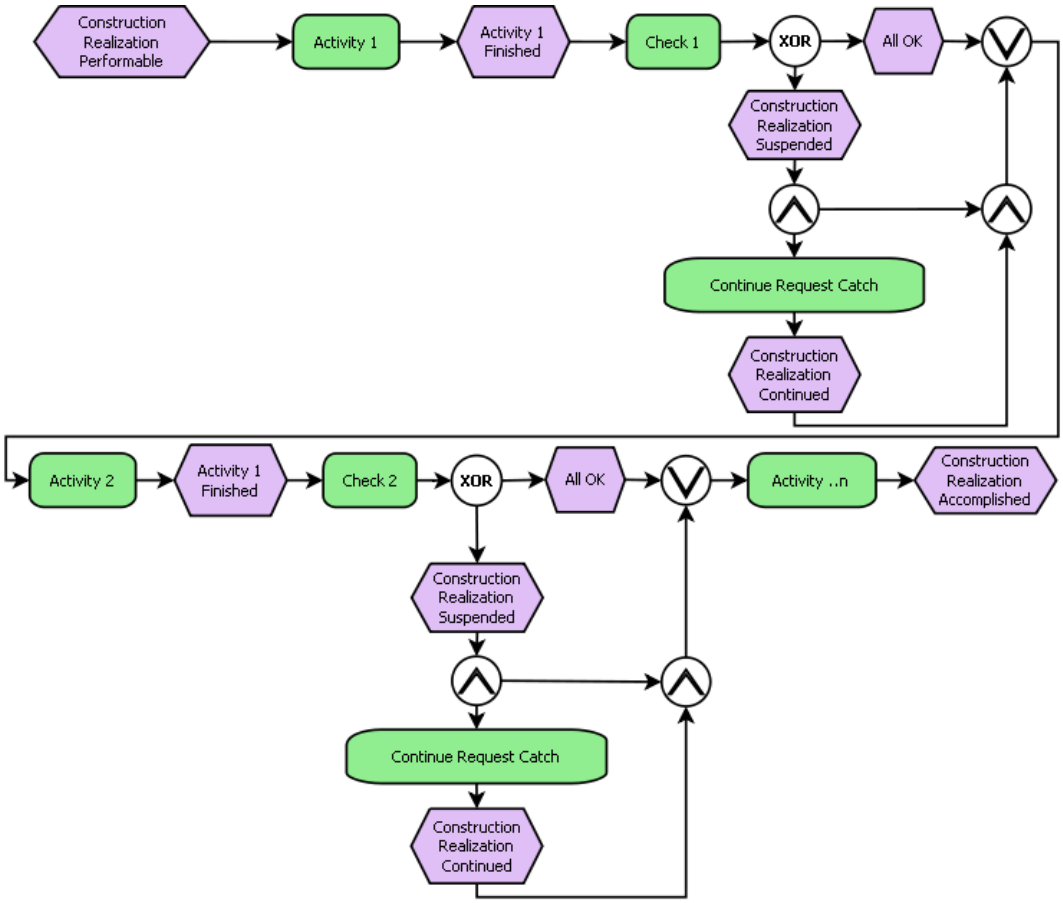

Figure 3-2: Suspension and Continuation Implemented through a Sub-Process in EPC

Suspension and continuation of an activity is not natively supported by the EPC, instead we have to help ourselves with general events that should have an effect on the running Construction Realization activity. This effect has to be incorporated by implementing it in the Construction Realization activity sub-process (Figure 3-2). The association of the Construction Realization activity and suspend and continue events is not captured by some visible relation - one has to look into the Construction Realization activity sub-process.

The capturing of suspending and continuing by a sub-process is not precise since the suspend and continue events may occur while any of Activity 1..n is running. Trying to make it more precise would force us to model sub-processes of Activities 1..n and so on. That is not a feasible way since even at this level the sub-process model is too complicated.

3.2 Model in BPMN

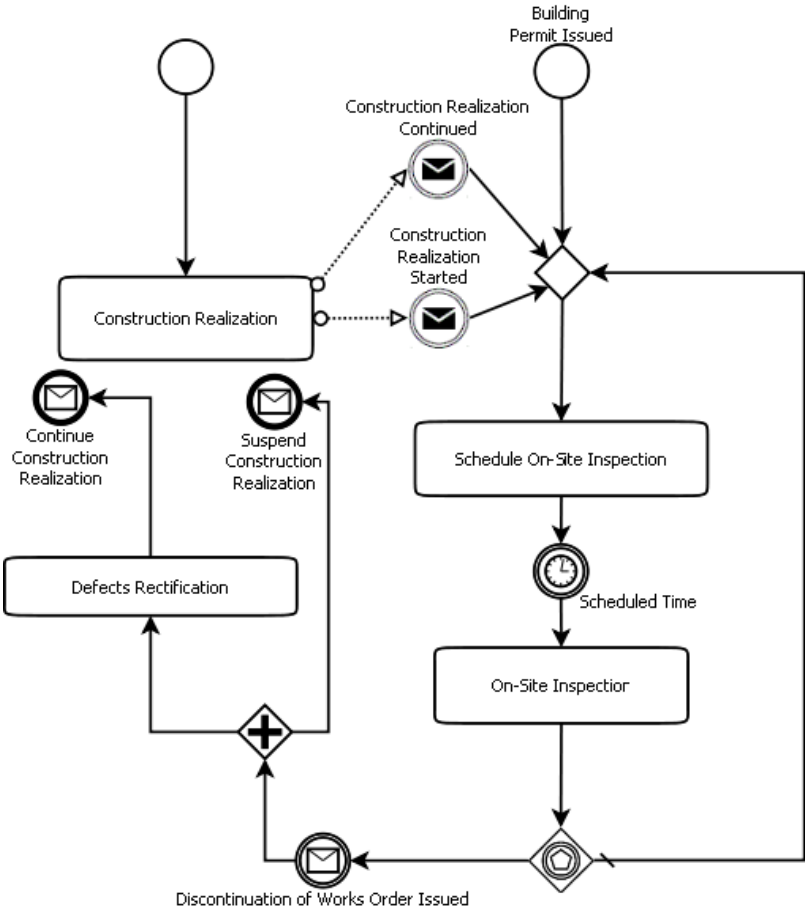

Figure 3-3: Defects Rectification in BPMN

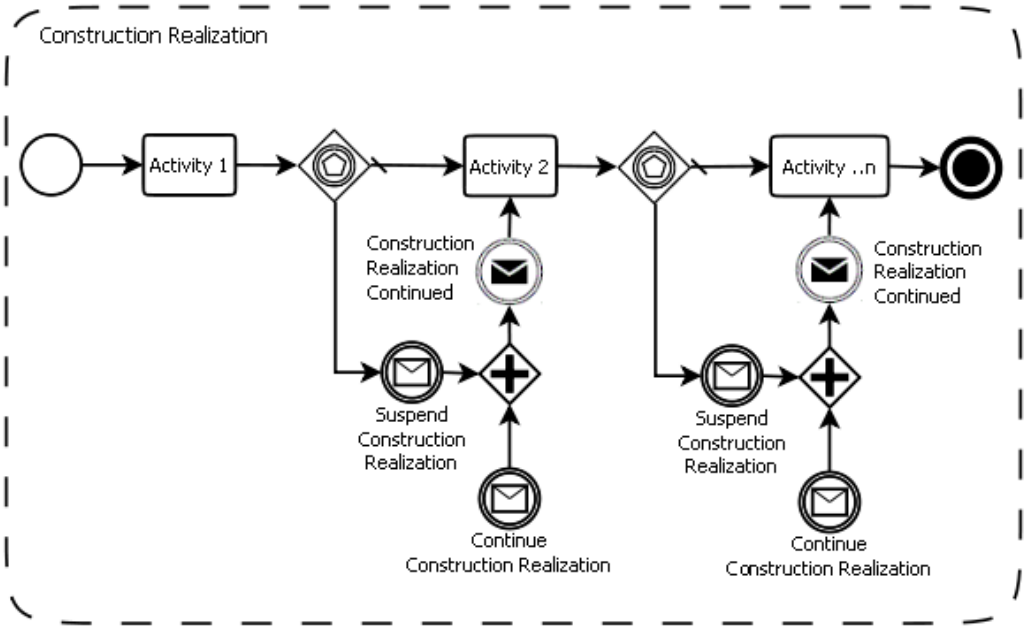

Figure 3-4: Suspension and Continuation Implemented through a Sub-Process in BPMN

The BPMN does not have specific construction for activity states *suspended* and *continued*; therefore we have to help ourselves with general events (in this case we used the message events). As we can see in in Figure 3-3 and Figure 3-4 this construction is not perfect.

First of all the association of the Construction Realization activity and suspend and continue events is not captured by some visible relation, one has to look into the Construction Realization sub-process.

Second, the capturing of the suspension and continuation is not precise as it may occur anytime during the Construction Realization performance. This forces us in BPMN to model the individual activities contained in Construction Realization sub-process (Figure 3-4). Here we face the same problem as before since the suspend and continue events may occur while any of Activity 1..n is being performed. Trying to make it more precise would force us to model sub-processes of Activities 1..n and so on. That is not a feasible way since even at this level it makes the sub-process model too complicated.

3.3 Impact Analysis Conclusions

The suspending and continuing construction realization example proved to be troublesome. None of the two process modeling languages, we model the example in, supports these activity states natively and therefore there had to be workarounds implemented. The solution presented for each modeling language is not perfect. Besides it creates an enormous overhead in the model, they also do not capture the substance of suspending and resuming correctly. It is not possible to suspend an activity immediately when the event, which is the cause of the suspension, occurs but the suspension has to wait until sub-activity of the modeled activity finishes and the decision about the suspension can be made.

4. Process State Diagram

The problems described in Chapter 3.3 can be overcome by modeling in process state diagram (PSD) - process state oriented modeling language proposed in (Svatoš, 2011).

The goal of PSD is to be as simple to understand as the EPC is and yet to be powerful enough to be able to capture clearly all the process events and states. This language is developed as process modeling language for reengineering and its main purpose is the description (Mayer, et al.,1995, p. 4) of modeled process not its execution.

PSD recognizes three types of process events and states: activity related, object related and time limit or point related.

Time limit and time point have defined in PSD their standardized lifecycles (Figure 4-1). The time point lifecycle is simple. First it is *set* the time that represents point in time. This can be either static value which is the same for all instances of this time point or it is set dynamically by some activity making possible for every instance of this time point to have different point in time set. When the set point in time occurs, the state of the time point changes to final state *occurred*.

The time limit has its lifecycle bit more complicated. The *set* state has the same definition as in the time point case. Immediately as the timer is started, the state of the time limit changes to *running (started)*. When running it can be *suspended* and later on *continued* or it may finish. When the time limit is *running* or *suspended* it can be also *reset* and then it waits to be started again.

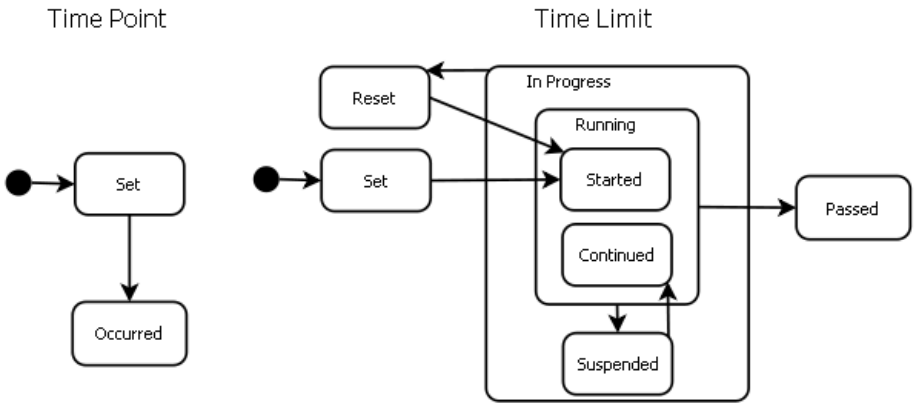

Figure 4-1: PSD Time Point and Limit Lifecycles

Object lifecycle is unique to each type of an object and situation and so there is no standard lifecycle. These have to be defined for each modeling case.

Activity lifecycle is defined in PSD on basis of the Figure 2-1 and there is introduced a lifecycle of a business activity which is captured in Figure 4-2.

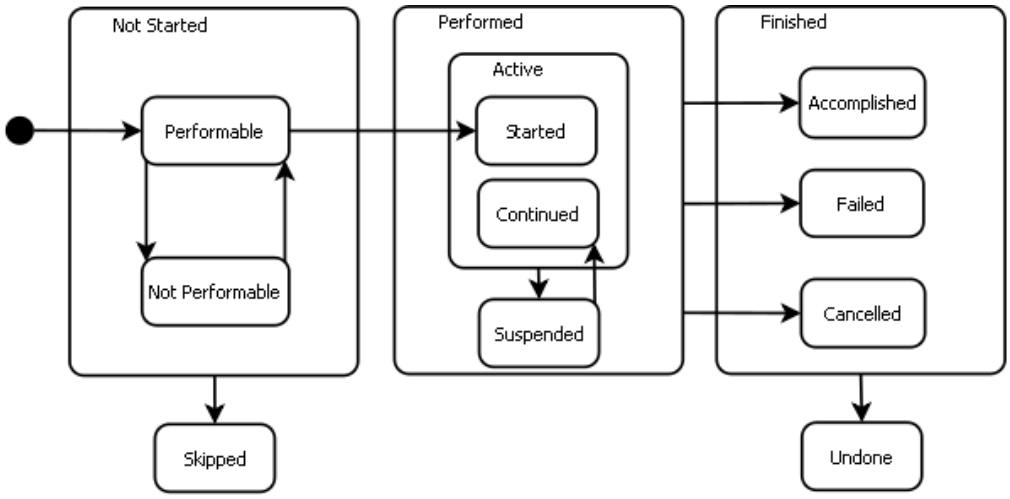

Figure 4-2: PSD Activity Lifecycle

Most of the changes in comparison to Figure 2-1 are done only in terminology, but there are some changes in structure done too. The *init* state was removed since it is valid only in computer software. In addition there is no *not running* state generalization as it is no longer necessary, since the s*uspended* state is now together with *running* state (*active*) detail of state *performed. Active* state differentiates whether the activity was *started* or *continued*. *Performable* and *not performable* states are just renamed *ready* and *disabled* states. The generalizing *closed* state was removed since it has no added value for this lifecycle diagram. Last change is possibility to change from state *suspended* to

any of finished states. Outside technical world it is not necessary to wait for an activity to become active in order to cancel it or finish it otherwise.

Sequences of process states are in PSD captured by precedence links. **Precedence link** depicts what combination of process states precedes the subsequent state. The subsequent process state occurs when all the conditions of the precedence link are fulfilled.

Precedence link is represented by an oriented graph where nodes are forks, conditions and a final node represented by a goal process state (subsequent state). The conditions are logical expressions, which are represented by an oriented graph where nodes are joins and process states (representing state conditions). Starting nodes and the goal node can be only process states.

There are two types of forks:

- **Exclusive fork** represents selection of one precedence link, which will be further evaluated, out of several possible precedence links (which share conditions defined before the exclusive fork) based on evaluation of condition of each outgoing precedence link, which is represented by a first join next to the exclusive fork.

  Each outgoing link has to have a condition next to the exclusive fork with one exception. There can be one outgoing link without a condition which would represent precedence link that would be activated in case all conditions of the other precedence links are evaluated as false.[3]

- **Parallel fork** represents split at the end of common part of conditions for several precedence links into individual precedence links. Other words, outgoing precedence links share the conditions defined before the parallel fork.

And there are three types of joins:

- **AND** represents condition which is evaluated as true when all incoming conditions are true.
- **XOR** represents condition which is evaluated as true when only one of the incoming conditions is evaluated as true.
- **OR** represents condition which is evaluated as true when one or more of the incoming conditions are evaluated as true.

4.1 PSD Notation

Figure 4-3 presents all elements of PSD. Notation is based on EPC modeling language since it is very close to the nature of PSD and this also allows us to use the extended set of business oriented elements used in eEPC (IDS Scheer AG, 2010) like Organization, Application system etc. within the PSD models.

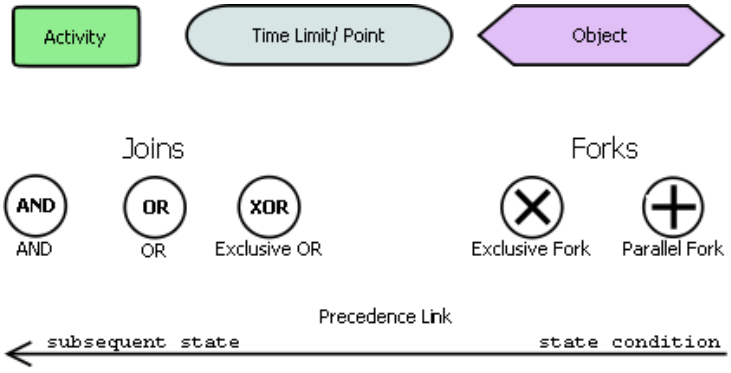

Figure 4-3: PSD Elements

The precedence link in PSD is similar to sequence flows in other process modeling languages. The difference is that the states are in PSD explicit. Figure 4-4 presents the most common sequence flow

[3] *In other words it represents an else flow*

in all process modeling languages. The explicitness of states allows the PSD to use precedence links for other sequences then just *accomplished ->started*.

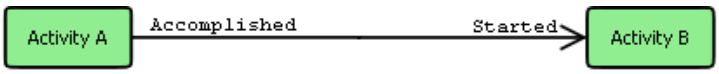

Figure 4-4: PSD Simple Precedence Link

4.2 Real Example Modeled in PSD

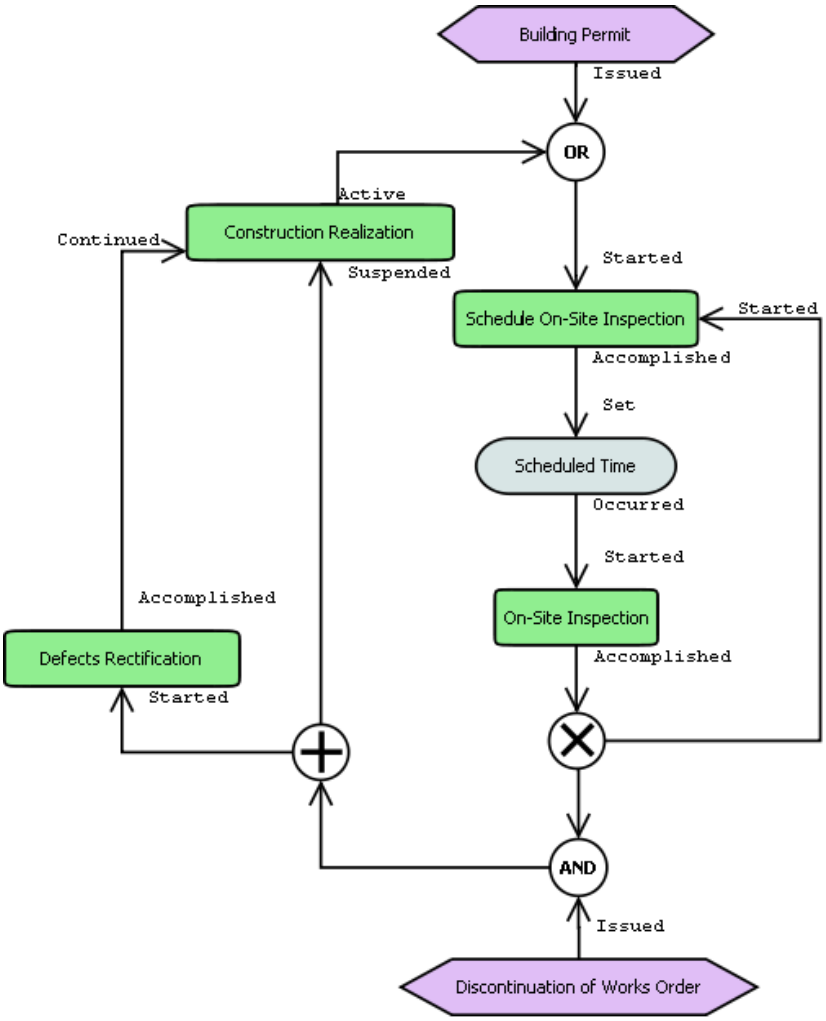

Figure 4-5: Defects Rectification in PSD

Figure 4-5 captures the example using the PSD. Suspending and continuing an activity is in PSD a simple operation. There is also captured very clearly the relation of the *continued* and *suspended* states to the Construction Realization activity. This solution does not suffer from necessity to model further sub-processes in order to be able to capture the possibility of an activity being suspended or continued and it also captures the fact that suspension and continuation can happen anytime while the Construction Realization activity is active.

5. Conclusions

In this work we have shown the issues the analyst encounters when modeling suspension and continuation of a process and provided solution, which can removes these issues.

We have started with activity lifecycle analysis on basis of research of prof. Weske and the analysis of two widely used process modeling languages, which shows that activity lifecycle is in reality much more complex than the contemporary process modeling languages think of them since they do not recognize the states before an activity is started and also the suspended, undone and skipped states from further stages of its life.

For this reason we have focused on defining of a real example focused on process suspension and continuation and capturing it in the two widely used process modeling languages in order to analyze how they can substitute their lack of complete activity lifecycle support.

The example illustrates in the models with what precision and clearness can the two used process modeling languages capture the process states and the analysis of the real example models shows that there are difficulties that make capturing suspension and continuation of a process hard.

The not native support of the suspension was solved by workarounds, but there should be noted that even though there are workarounds possible, the models themself show that there is a significant cost for this kind of solution and the result is not perfect. Every workaround produces overhead that makes the model more complicated – it is not only a problem of number of entities but also usage of sub-processes, which are sometimes due to activity complexity almost impossible to capture. Decomposition of unstructured activities or very complex activities forces the analysts to go into far greater detail just for capturing situation that could be captured at the higher detail level that is appropriate for the situation description. And yet capturing such situations like that an activity can be suspended anytime remains beyond expressive power of the contemporary process modeling languages.

As a solution to the difficulties presented above we have presented the process state diagram (PSD), which provides the ability to capture the whole range of activity states defined in this work.

We illustrate the capabilities of PSD by modeling the example and discussing the advantages the PSD brings in cases where the contemporary process modeling languages had problems or their usage required a lot of effort to capture the example correctly. There are no workarounds necessary and the level of detail is driven by the analyst needs and not by the process modeling language constructions.

6. Bibliography

Act 183, 2006 on town and country planning and building code (Building Act). *Collection of Laws of the Czech Republic. Part 63*

Act 500, 2004 on Administrative Procedure Code. *Collection of Laws of the Czech Republic. Part 174.* Prague : s.n., 2004, pp. 9782-9844. http://aplikace.mvcr.cz/sbirka-zakonu/ViewFile.aspx?type=c&id=4478.

Act No. 42/1994, on State-contributory supplementary pension insurance. *Collection of Laws of the Czech Republic. Part 14.* 1994

Becker, J., et al., 2010. *Exploring the Status Quo of Business Process Modelling Languages in the Banking Sector – An Empirical Insight into the Usage of Methods in Banks.* s.l. : 21st Australasian Conference on Information Systems (ACIS 2010), http://aisel.aisnet.org/acis2010/8/

Civil Procedure Code, 1963. *Collection of Laws of the Czech Republic. Part 56*

*Flexible Mortgage*2011. [Online], http://www.kb.cz/en/people/individuals/flexible-mortgage.shtml

IDS Scheer AG, 2010. *ARSI Method: ARIS Platform Version 7.1 – Service Release 8.* [Online] http://documentation.softwareag.com/aris/aris71e/method_manual_aris_s.pdf.

Mayer, R.J., et al., 1995. *Information Integration For Concurrent Engineering (IICE) IDEF3 Process Description Capture Method Report.* s.l. : Knowledge Based Systems, Inc., http://www.idef.com/pdf/Idef3_fn.pdf

Mendling, J., Neumann, G. and Nüttgens, M., 2005**.** Yet Another Event-Driven Process Chain. BPM'05 Proceedings of the *3rd international conference on Business Process Management* Springer-Verlag Berlin, Heidelberg, Vol. 3649, pp. 428-433

Object Management Group, 2011a. UML Version 2.4 - Beta 2. *Object Management Group.* [Online] http://www.omg.org/spec/UML/2.4/

Object Management Group, 2011b. Business Process Model and Notation (BPMN) Specification Version 2.0. [Online], http://www.omg.org/spec/BPMN/2.0

Řepa, V., 2012, *Information Modeling of Organizations.* s.l. : Bruckner, 2012. ISBN: 978-80-904661-3-5

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

Weske, M., 2007**.**. *Business Process Management: Concepts, Languages, Architectures.* s.l. : Springer, ISBN: 3-540-73521-6

Yourdon, Ed. 2006. *Just Enough Structured Analysis* - rev. 051406. [Online] http://www.yourdon.com/jesa/jesa.php

JEL Classification: D8, M1

This article should be cited as:

Svatos, O., 2012. Modeling Suspension and Continuation of a Process. *Journal of Systems Integration* 3 (2), pp. 62 - 73 [Online] Available at: http://www.si-journal.org. ISSN: 1804-2724