Preprint Version 1

The Process Checklist – Simple Establishment of Execution Support for Human-driven Processes

Transforming Process Models to Process Checklists

Michaela Baumann · Stefan Schönig · Michael Heinrich Baumann · Stefan Jablonski

2015

Abstract In traditional approaches business processes are executed on top of IT-based Workflow-Management Systems (WfMS). The key benefits of the application of a WfMS are task coordination, step-by-step guidance through process execution and traceability supporting compliance issues. However, when dealing with human-driven workflows, conventional WfMS turn out to be too restrictive. Especially, the only way to handle exceptions is to bypass the system. If users are forced to bypass WfMS frequently, the system is more a liability than an asset. In order to diminish the dependency from IT-based process management systems, we propose an alternative way of supporting workflow

M. Baumann
Databases and Information Systems
University of Bayreuth
Tel.: +49-921-55-7625
Fax: +49-921-55-7622
E-mail: michaela.baumann@uni-bayreuth.de

S. Schönig
Databases and Information Systems
University of Bayreuth
Tel.: +49-921-55-7627
Fax: +49-921-55-7622
E-mail: stefan.schoenig@uni-bayreuth.de

M. H. Baumann
Applied Mathematics
University of Bayreuth
Tel.: +49-921-55-3280
Fax: +49-921-55-5361
E-mail: michael.baumann@uni-bayreuth.de

S. Jablonski
Databases and Information Systems
University of Bayreuth
Tel.: +49-921-55-7620
Fax: +49-921-55-7622
E-mail: stefan.jablonski@uni-bayreuth.de

execution that is especially suitable for human-driven processes. We introduce the so-called process checklist representation of process models where processes are described as a paper-based step-by-step instruction handbook.

Keywords process modelling · process checklists · paper-based process execution

1 Introduction

This article is an extended paper of the conference proceeding [1]. In addition to various improvements and further concepts the work at hand extends our previous article by a description of the actual implementation as well as by a detailed evaluation and case study section.

Since approximately 20 years process management is regarded as an innovative technology both for the description of complex applications and for supporting their execution [7]. In traditional approaches business processes are executed on top of IT-based Workflow-Management Systems (WfMS) [17]. The key benefits of the application of a WfMS are task coordination, step-by-step guidance through process execution and traceability supporting compliance issues [14]. However, when dealing with human-driven workflows that heavily depend on dynamic human decisions, conventional WfMS turn out to be too restrictive [15]. Especially, the only way to handle exceptions – which regularly occur in human-driven workflows – is to bypass the system. If users are forced to bypass WfMS frequently, the system is more a liability than an asset [15]. In total, users start to complain that "the computer won't let them" to do the things they like to accomplish [2]. So users like to get more independent from "electronic systems" in order to become more flexible. If original documents are needed for executing a process, in many cases a paper-based execution model is preferred [9].
Furthermore, the introduction of a WfMS is regarded as a huge, cost-intensive project [10]. Many organizations cannot afford to introduce such a system therefore. However, they desire to manage their processes since they regard them as valuable and effective. In order to diminish the dependency from IT-based process management systems, we propose an alternative way of supporting workflow execution that is especially suitable for human-driven processes, like it is the case for example in public administration and authorities. We introduce the so-called process checklist representation of process models. Here, processes are described as a paper-based step-by-step instruction handbook. The process checklist is handed over during process execution from process participant to process participant.
Successful task accomplishments are recorded through signatures of corresponding agents. In principle the most important statement is that at the end of the process all signatures are on the checklist. So it is completely output oriented. Nevertheless, the checklist method describes a valuable form of

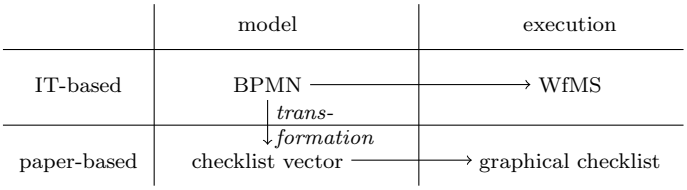

Fig. 1: Schematic approach to differentiate paper-based from IT-based process management systems.

process usage and widens its spectrum towards non-computer based and extremely flexible process execution. If needed, certain parts of the checklist can even be deleted, changed or added during the execution by simply using a pen. However, traceability is still maintained. Besides, the process checklist also supports the key benefits of traditional WfMS. The checklist is handed over to responsible agents (task coordination), process tasks are serialized and marked by a unique identifier (step-by-step guidance) and the checklist itself as well as the corresponding signatures ensure traceable process execution. The work at hand provides the general structure of process checklists as well as an elaborate transformation algorithm of basic BPMN process model elements [11] to process checklists. Fig. 1 shows a comparison of traditional IT-based process execution and the paper-based approach provided by the work at hand.

2 Background and Related Work

A checklist is a list of items required, things to be done, or points to be considered, used as a reminder [16]. Checklists are generally seen as a suitable means for error management and performance improvement in highly complex scenarios like clinical workflows [5]. Therefore, we propose to define a generic method for transforming general process models to the checklist representation. The problem of transforming a model drawn in one business process modeling notation into another notation has been examined in different papers, e.g., [6], [8]. However, to the best of the authors' knowledge, the transformation of process models to a checklist representation has not been discussed so far, except for the previously mentioned paper [1] which is the basis for the work at hand. Before specifying the transformation of process models into checklists, we have to determine how suitable process models should look like and what elements a checklist consists of. These specifications are necessary to give concrete mapping rules. For process models, only basic elements of the Business Process Modeling Notation are allowed, as [18] shows this is enough in most cases and as the paper at hand has to be seen as a first approach to this topic.

Definition 1 (Process model) A process model is defined according to BPMN 2.0 (see e.g. in [11]) allowing for the following basic elements:

- flow objects: activities, events (start, end), gateways (AND, XOR, OR)
- sequence flows
- data (input/output) objects
- participants: one pool, possibly separated into different lanes

As we consider the application of checklists appropriate only within one company, there should not occur processes with more than one pool. Therefore, we do not have to take message flows into account. Which forms of activities, events and gateways can be covered with our transformation rules will become apparent when it comes to the concrete transformation of process models into checklists. We specify a checklist as follows.

Definition 2 (Checklist vector) A checklist is a vector $\mathcal{C} = (p_1^t, p_2^t, \ldots, p_n^t)$, $n \in \mathbb{N}$, $t \in \{o, c\}$ with two different kinds of components:

$$p_i^o = (ID_i, AC_i, OD_i, AG_i)$$

with ID_i, AC_i, OD_i, AG_i being strings and

$$p_j^c = (AN_j, CO_j, GT_j, AG_j)$$

with AN_j, CO_j, AG_j being strings and GT_j being a vector of the form

$$GT_j = (s_j, a_{j,1}, g_{j,1}, a_{j,2}, g_{j,2}, \ldots, a_{j,k}, g_{j,k})$$

with $k \in \mathbb{N}$, strings $a_{j,l}$, integers (or NULL) $g_{j,l} \in \{1, \ldots, n\} \cup \{NULL\}$, $l = 1, \ldots, k$, and $s_j \in \{0, 1\}$.

This definition uses a lot of different variables that need some explanation: The component p^o of a checklist vector as defined above is called operating point. It contains information about incoming data objects (ID), the activity (AC) which may be an activity in the literal sense of BPMN or an event, outgoing data objects (OD), and the performing agent (AG). An operating point gives more or less conrete instructions to the respective agent about what he has to do. The other component of a checklist vector, p^c, is called control point. In general, a control point is a transformed gateway, therefore it contains information about the condition (CO) which may also be empty if it corresponds to a parallel gateway, and the responsible agent (AG as in p^o). AN is a component kept free for special annotations (we will see examples later) and GT is a vector with one boolean component s and k pairs of string (a) and integer (g) components. g refers to other components of $\mathcal{C}$ and is therefore element of $\{1, \ldots, n\}$ or $NULL$.
With this formal definition of a checklist, the checklist vector, it is already possible to give concrete mapping instructions as listed in the next section. Before we turn towards this subject, we want to give the reader a visual impression of how the two components p^o and p^c may be illustrated on a graphical checklist in Fig. 2 and Fig. 3. The components of vector GT_j are shown in Fig. 4.

In which way these checklist components are filled with information given by the process model and how the resulting operating and control points are represented in the graphical checklist is explained in the next two sections.

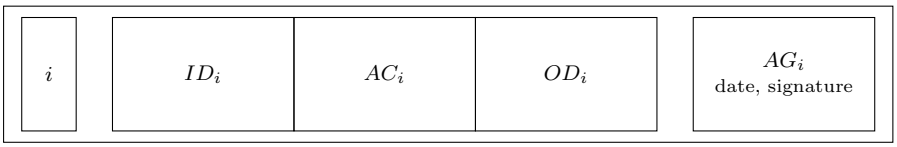

Fig. 2: Visualization of p_i^o which means the i-th component of $\mathcal{C}$ is an operating point.

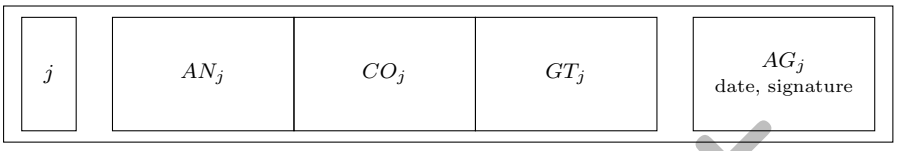

Fig. 3: Visualization of p_j^c which means the j-th component of $\mathcal{C}$ is a control point.

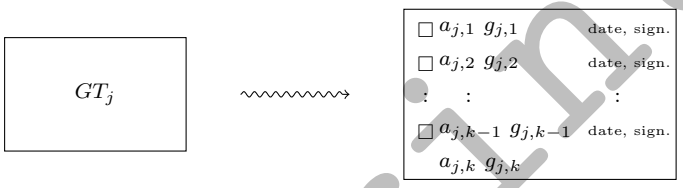

Fig. 4: Visualization of GT_j. Entries date and signature in the third column only appear, if $s_j = 1$. The k-th row never has a square in the first column nor a date and signature. In fact, the k-th row may be empty, i.e., $a_{j,k} =$ "" and $g_{j,k} = NULL$.

3 Transformation of Process Model Elements

This section focuses on generating a checklist, that means it is explained, in which way the single elements of the (BPMN) process model are transferred into either operating points or control points. These steps are basically performed in a simply algorithmic way, except for parallel gateways.

3.1 Transformation of Activities

Activities are transformed straight into operating points p^o. Their description is mapped on the field AC whereas all directly incoming data and directly outgoing data is mapped on the field ID and OD respectively. The participant of the corresponding lane or hierarchy of lanes, that may, e.g., be a single person is mapped onto the field AG. An example of an activity with documents and participant is given in Fig. 5.

3.2 Transformation of Subprocesses

Occurring subprocesses, marked with a symbol as seen in Fig. 6, may be taken into a checklist in different ways:

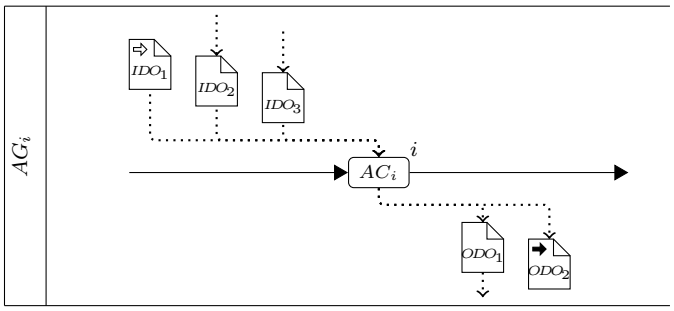

Fig. 5: Exemplary excerpt from a process model with labels according to an operating point p_i^o. $ID_i = IDO_1.IDO_2.IDO_3$ and $OD_i = ODO_1.ODO_2$.

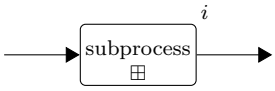

Fig. 6: Symbol for a subprocess in BPMN 2.0.

1. Include the complete subprocess. This leads to a comparatively long but correct checklist.
2. Generate a new checklist for each subprocess. One control point j has to be inserted into the original checklist with work instructions for printing and passing on the new checklist ($a_{j,1}$ ="print and pass new checklist x to agent y", $g_{j,1} = NULL$) which has to be signed due to $s_j = 1$ and with instructions for waiting for this checklist to come back completely processed. Parameter $a_{j,2}$ is set to "finally go to" and $g_{j,2} = j + 1$.

3.3 Transformation of Gateways

3.3.1 Transformation of Exclusive Gateways

An exclusive split gateway (Fig. 7) has to be transformed into a control point in which the decision question and the possible answers with the respective "go to"-numbers ($g_{j,1}, g_{j,2}$ and $g_{j,3}$ in Fig. 7) are mentioned. Parameter s_j is set to 0 as the decision has not to be signed in field GT_j. If there is an exclusive join gateway (Fig. 8) too, at the end of each branch of the respective splitting gateway a jump instruction to the next point in the checklist after the join gateway ($g_{j,4}$ in Fig. 8) must be inserted, except the next point following a branch is the point following the join gateway. The execution of an exclusive gateway may cause problems if at least one "go to"-number is in the past, but this problem will be solved in the next section.

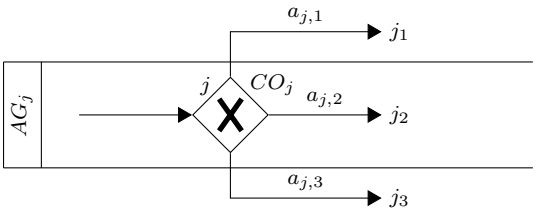

Fig. 7: Exclusive split gateway with question CO_j and possible answers $a_{j,1}, a_{j,2}, a_{j,3}$. p_j^c: AN_j may be used for data. $g_{j,k} = j_k$, $k = 1, 2, 3$, $a_{j,4} =$ "" and $g_{j,4} = NULL$.

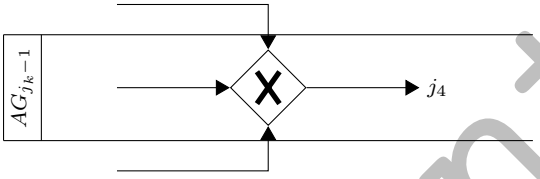

Fig. 8: Exclusive join gateway that does not have to exist if the outgoing branches of the exclusive split gateway end with terminal events. $p_{j_k-1}^c$: $AN_{j_k-1} =$"", $CO_{j_k-1} =$ "$XOR\,end$", $s_{j_k-1} = 0$, $a_{j_k-1} =$ "$goto:$", $g_{j_k-1} = j_4$, $k = 2, 3$.

3.3.2 Transformation of Parallel Gateways

There are several ways of transforming parallel gateways into a checklist whereby all of them have different advantages and disadvantages. Some of these possibilities are listed below. Note, that a mixture of these transformation possibilities is also conceivable.

Static Sequential Transformation This type of transforming a parallel gateway takes the several branches of the process model that are between the split and join gateway and brings them into an arbitrary order. The gateway itself is not mapped to the checklist.

Dynamic Sequential or Postbox Transformation A parallel split will be transformed to a control point p_j^c. The parallel branches in the process model have to be written down in a sequential way in the checklist. At the end of each branch a jump to p_j^c, realized with a simple control point, is necessary and in p_j^c the number of the point following the respective parallel join has to be noted. The parameter specifications are in the captions of Fig. 9. There are different ways of executing this parallel split, and some of them correspond to another transformation, but this will be dealt with in the next section.

Parallel Transformation For each parallel branch a checklist is generated and distributed by the agent of the split gateway (AG_j in Fig. 9) to the agents of the first process element of the branches. It is modelled as one control node p_j^c. If the gateway splits into k branches, then $a_{j,k+1} =$"Finally go to" and $g_{j,k+1} =$

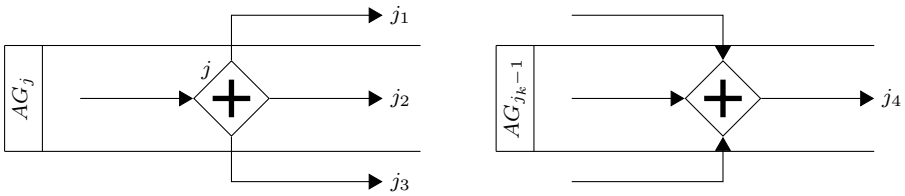

Fig. 9: Parallel split gateway p^c_j: AN_j for annotation, e.g. data objects, CO_j = "AND", $s_j = 1$, $a_{j,1}, \ldots, a_{j,3}$ ="", $g_{j,1} = j_1$, $g_{j,2} = j_2$, $g_{j,3} = j_3$, $a_{j,4}$ ="Finally go to", $g_{j,4} = j_4$. Parallel join gateway $p^c_{j_k-1}$: AN_{j_k-1} ="", CO_{j_k-1} ="AND end", $s_{j_k-1} = 0$, a_{j_k-1} ="go to", $g_{j_k-1} = j$, $k = 2, 3, 4$.

$j+1$. If the name of the current checklist is "Checklist", then CO_j ="AND – print checklists "Checklist_sub1",...,"Checklist_sub$k$", if the names of the sub-checklists are "Checklist_sub1",...,"Checklist_subk". Of course $a_{j,1}, \ldots, a_{j,k}$ have to reference these sub-checklists, $g_{j,1}, \ldots, g_{j,k} = NULL$ and $s_j = 1$ which means that signatures for all returning sub-checklists are needed (see also procedure for subprocesses in Section 3.2).

3.3.3 Transformation of Inclusive Gateways

The transformation of inclusive gateways can be done similar to the transformation of parallel gateways. More precisely, there are the possibilities to use the dynamic sequential or postbox transformation or the parallel transformation. The only difference is, that in p^c_j we have CO_j and $a_{j,1}, \ldots, a_{j,k}$ like in the exclusive gateway transformation, i.e., the condition/question and the answers have to be taken over from the process model.

3.4 Transformation of Events

3.4.1 Direct Transformation of Events

Some events, like signal events, can be transformed like activities, that means to p^o_i, with AC_i = "" or AC_i is used for transmitting some message.

3.4.2 Indirect Transformation of Events

Most events, like time, condition and message events, are requirements for the next point in the checklist and can be modelled this way. This requirement is written down in AC or AN of the following operating or control point.

3.4.3 Ignored Events

Other events, like the start event, can be ignored, that means they have no respresentation in the checklist, because they won't influence the execution.

3.5 Transformation of infrequent, mutually exclusive activities or branches

As a paper-based checklist needs direct human treating during its execution (see Section 4), it can be handled very flexible by the agents, which does not mean, that the agents can do what they want during the execution, but they have a certain freedom which is not given when processing with the aid of a WfMS. Consider a clinical workflow where a diagnosis (outgoing data) has to be made in one step and, according to this diagnosis, the treatment has to be executed in the following step (diagnosis as incoming data). Implementing all different kinds of diagnosis-depending treatments would cause an exclusive gateway with nearly innumerable branches or subprocesses, not to mention that all these eventualities have to be considered at modeling time (cf. [3]. What if a certain treatment has been forgotten because of its rareness, for example?
When facing this problem in the context of checklists, the following solution is conceivable: List only the most frequently made diagnoses in the corresponding XOR-control point $((a_{j,1}, g_{j,1}), \ldots, (a_{j,l}, g_{j,l}))$ and add one $(a_{j,l+1}, g_{j,l+1})$ with $a_{j,l+1} = \mathit{other}$ and $g_{j,l+1}$ referring to an empty operating point where the conrete diagnosis and all incoming and outgoing data can be entered at run time by the doctor in charge. These empty operating points offer a way to reduce complexity of the process model and to prevent the process to get stuck during its execution. But as they require good knowledge about the process they can only be filled in by agents with the corresponding expertise and should therefore not be overused.

4 Enactment of the graphical checklist

A graphical checklist contains a cover sheet with name of the checklist (name of the process), timestamp, and a list for writing down the current checklist and the current point, i.e., the next point to be worked on. Furthermore, a graphical checklist consists of at least one checklist as described above (resulting from a checklist vector) with a consecutive number, starting with 1, a receipt book and a list for data objects and maybe data objects (documents). An illustrating example for the cover sheet and a graphical checklist with consecutive number 2 is given in Fig. 10.

When starting a process with checklists, the "process owner", i.e., that person starting the execution of the process, has to print the checklist with cover sheet and data object list. Then he assigns the checklist its current number 1. Input data, that means input documents, have to be added and scheduled in the respective list. On the cover sheet "1–1" is noted, that means, the current status of execution is "checklist 1" and "point 1". In addition, he has to write his name on the cover sheet so that the checklist can be handed over to him after finishing the process. This graphical checklist has to be passed to the agent named in point 1, who has to check for completeness, that means especially if all listet documents are handed over, and quit the delivery. The process owner

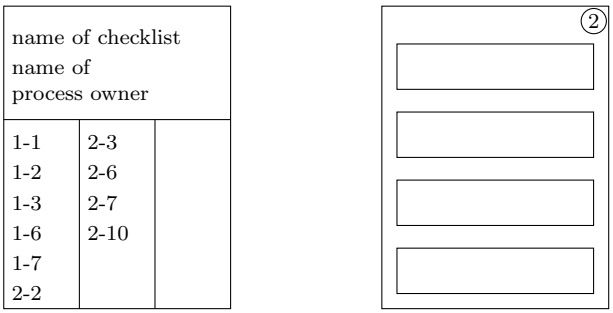

Fig. 10: Cover sheet (left-hand side) with name of the checklist/process, name of process owner and list of the next points to be executed; checklist (right-hand side) with current number in the upper right corner and operating/control points. Obviously, in checklist no. 1 a gateway caused a jump into the past (from point no. 7 to point no. 2).

has to archive the signed receipt for later reconstruction if necessary.

Every time an agent gets the graphical checklist he has to run through this acknowledgement process (checking the documents for completeness, sign a receipt) and then check for the current point of the checklist on the cover sheet. When the last entry is 1–23 he has to look at point 23 of the current checklist, that has number 1, and execute this point, if all necessary documents are available and possible conditions are fulfilled. Of course, the agent named in this point should be correct (otherwise the checklist has not been handed over properly). After execution of the current point he has look which agent is next. If it is himself he executes the next point and writes it down on the cover sheet, else he updates the document list, writes the next point on the cover sheet, hands the checklist over and archives the received receipt. If one agent sends a document directly to another, this document has to be deleted from the data object list and maybe listed again later on by the other agent.

4.1 Execution of Operating Points

Operating points are executed straightaway as described above, performing the task (with possible constraint resulting from a transformed event) as given in *AC*. If documents are produced, they should correspond to that ones listed in the outgoing documents *OD*. After performing the task, he signs the operating point for making clear, he has finished this point.

4.2 Execution of Control Points

4.2.1 Execution of Exclusive Gateways

If a control point resulting from an exclusive gateway has to be processed, the agent has to check for the condition or question in field *CO*. He marks

his answer in GT in the box $\square$ in front of the corresponding answer $a_{\cdot,l}$. If there are any documents helping him to decide, they are listed in AN. After marking he gets the number of the next point, $g_{\cdot,l}$. Two possible sceneries may occur: $g_{\cdot,l}$ is greater than the current point number, then everything can go on as before. If $g_{\cdot,l}$ is smaller than the current point number, then there is a problem, as that point with number $g_{\cdot,l}$ may have been processed already in the past and therefore is signed already. If such a return occurs, than the agent of the control point has to print a new checklist (just the checklist itself) and assign it the number $i + 1$ if the number of the current checklist was i. On the cover sheet, he writes for the next point to be executed $(i + 1)$–$(g_{\cdot,l})$. After doing this, he signs in field AG and passes the new checklist (together with the old one for reconstruction opportunity) to the agent of point $g_{\cdot,l}$. This agent has to recognize that the consecutive number of the checklist has changed which is obvious on the cover sheet.

4.2.2 Execution of Parallel Gateways

Static Sequential Transformation If a parallel gateway was transformed in the static sequential way, then it does not appear in the checklist, that means the performing agents do not know, that there has been a gateway in the BPMN process model. All branches are executed in the specific order as chosen by the person who transformed the process model.

Dynamic Sequential Transformation When coming to a control point being the transformation of a parallel gateway with the dynamic sequential method the agent of that point can decide about the execution order of the different branches during the processing of the checklist. He can take into account the current circumstances like availability of the agents in the different branches, or anything else. When he chooses one branch, he marks his decision in the corresponding box $\square$, notes it on the cover sheet and passes the graphical checklist over to the agent of the respective point, on the right-hand side of the marked box. The branch is processed and at the end there is a control point that refers back to the control point where the decision of the branch was made. So, the agent gets the checklist back (with checking for all documents and quitting again) and signs the chosen branch in GT (that one with the marked box, that has not been signed yet). Then he chooses the next branch to be processed the same way as before. If all branches have been marked and signed, then he signs the whole control point in field AG and passes the checklist over to the agent of that point listed after "finally go to" in GT. The whole procedure can be reconstructed with the notes on the cover sheet.

Postbox Method If parallel gateways are performed with the postbox method, the checklist itself looks the same as transformed according to the dynamic sequential way. The difference is in the execution, as the postbox method allows for parallel processing of the different branches. When the performance of a checklist reaches an AND control node the checklist is posted like an

announcement in one place together with all documents (that can be stored in a postbox) and all agents can look for the next points that have to be executed on the cover sheet, where all first points of the different branches have to be noted in a parallel way. With this method, the documents do not have to be handed over from one point to another. After finishing all branches, the agent of the control node that started the postbox method collects the checklist and all documents now being in the postbox, checks for completeness, signs in AG if everything is okay and goes on as before. This method may become confusing and needs initiative of all agents. But it considers the parallel aspect of parallel gateways.

Parallel Transformation With this method it is also possible to consider simultaneity of the different branches. The agent of the control node prints all required sub-checklists, marks the boxes □ in GT if handed over together with needed documents to the respective agents of the first points in the branches, as listed in GT, and signs every returning sub-checklist in GT. If all sub-checklists have returned, he signs in AG and the execution of the control node is finished. As one can imagine, this method is more elaborate, as multiple checklists have to be generated, but it provides a good overview over the process in contrast to the postbox method. We recommend this method if the branches are relatively long, so that the effort of generating more than one checklist is somehow justified.

The mentioned transformation and execution versions are somehow suggestions, clearly many other versions are imaginable and of course different versions can be mixed as we would probably suggest in the situation of Fig. 11. Here, the two long subprocesses $l1$ and $l2$ can be executed with two separate checklists whereas the short subprocess can be included into the checklist in a (dynamic) sequential way, i.e., it would be performed before or after the two long subprocesses.

4.2.3 Execution of Inclusive Gateways

Like the transformation of inclusive gateways, the execution of inclusive gateways can again be seen as a mixture of exclusive and parallel gateways. The agent of the corresponding control point has to choose his anwers (mark the boxes □), in contrast to exclusive gateways possibly more than one, and then for the chosen ones he can proceed like with parallel gateways (except for the static sequential method, as this was no possible transformation for inclusive gateways).

All methods mentioned so far require a well-modelled process model, that means for example, that there are no returns out of AND branches, no document is needed in parallel branches without having a copy of it, or that no document is archived if there is the possibility of a return into the past where this document will be needed again. Changes of the underlying process model

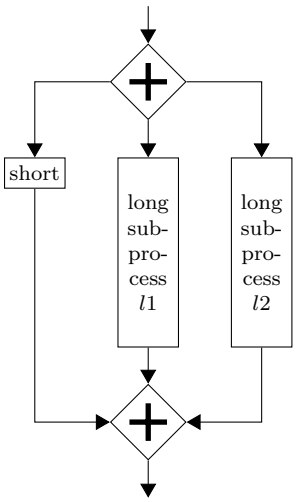

Fig. 11: Schematic part of a process model with one short and two comparatively long parallel subprocesses that can be executed as a composition of the presented methods.

involve modifications of the checklist for all future process instances. If problems or questions during the execution of one checklist occur, one should confer with the corresponding process owner.

5 Implementation

In order to provide a simple transformation possibility the described transformation procedures from BPMN to a process checklist representation have been implemented in a C-Sharp application using the Microsoft .NET framework. This way, process checklists can be generated from existing BPMN models in a few seconds.

5.1 Checklist Meta-Model and Model Transformation

In a first step, the user selects the BPMN-XML file to be transformed from a file dialog. Subsequently, the selected file is parsed by the application. As a result, an instantiation of the (simplyfied) BPMN meta-model is generated. Note, that for this approach we are only considering the basic BPMN elements as described in Section 2. Afterwards, the user has to choose the transformation method of possibly occurring parallel gateways, i.e., whether a static sequential, a dynamic sequential or a parallel transformation method is preferred. The provided information is finally used to transform the BPMN model instance to an instance of the checklist meta-model. The meta-model as an UML diagram is shown in Fig. 12. A checklist consists of several *ChecklistElements* that can either be an *OperatingPoint* or a *ControlPoint*. An OperatingPoint

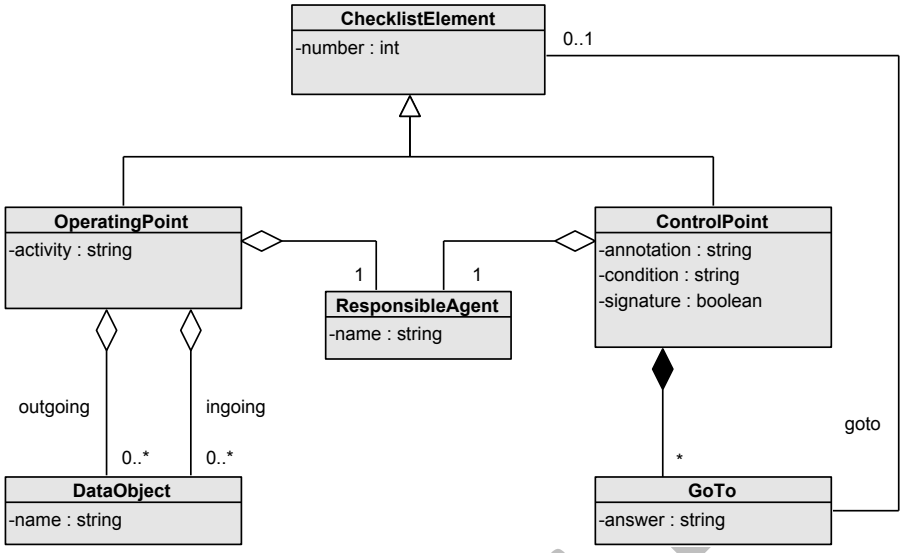

Fig. 12: Meta-model of the process checklist.

can have several ingoing as well as outgoing *DataObjects* and is performed by exactly one *ResponsibleAgent*. A ControlPoint has several *GoTo*-instructions and has also exactly one ResponsibleAgent. A GoTo-instruction refers to a ChecklistElement again.
The different checklist model elements can finally be read and visualized. Of course, there is no fixed checklist visualization method. However, the visualization method presented in the work at hand is the result of different case studies and discussions with process participants. It turned out to be an adequate and understandable representation.

5.2 Checklist Serialization

In order to be able to adequatly save and share generated checklists the application additionally provides a possibility to serialize checklists to XML[1]. The following listing shows an excerpt of the generated XML code of the example checklist vector of Fig. 14.

```
<checklist process='Subscribing for an exam'>
      ...
      <controlpoint no='2' annotation='' condition='Exam type?'
       signature='0'>
            <gotolist>
                  <goto answer='written' gotoNo='3' />
```

[1] Complete example checklist XML-files as well as a XML Schema definition can be found at the project website. See *checklists.kppq.de* for more information.

```
                <goto answer='oral' gotoNo='9' />
            </gotolist>
            <agent name='Student' />
      </controlpoint>
      ...
      <operatingpoint no='5' activity='Perform exam correction'>
            <ingoing>
                <DataObject name='Exam unmarked' />
            </ingoing>
            <outgoing>
                <DataObject name='Exam marked' />
            </outgoing>
            <agent name='Auditor' />
      </operatingpoint>
      ...
</checklist>
```

6 Evaluation and Analysis

For evaluating the desired advantages of paper-based checklists in contrast to the corresponding BPMN-based process models as execution support, the analyzed topics were divided into two categories, namely objective and subjective topics. The objective topics, flexibility, parallelism and length, could easily be derived from the execution support tools after determining the particular evaluation criteria. The subjective topics, comprehensibility, orientation and reliability, had to be derived from interviews and surveys and an ensuing statistical evaluation. Two different processes were modeled to get more reliable results. The first one, subscribing for an exam, was modeled as a graphical BPMN process model as shown at the end of the paper in Fig. 18 and a straight-forward checklist. Parts of the checklist are represented in Fig. 13. The corresponding checklist vector as basis for the graphical checklist is presented in Fig. 14. The graphical BPMN model was chosen as we discovered, that this is the usual way of supporting the execution of human-based processes, available for all involved actors, in small and middle-sized companies. A map of the graphical BPMN-model is posted publicly which allows several instances of the process to proceed simultaneously. An IT-based WfMS is often not available or too expensive for these firms. The first process included the student, the chair's secretariat, the auditor and the assessor as involved agents. For the second process, applying for a business trip, again a graphical BPMN process model was generated as well as two different checklists to examine the differences presented in Section 3.3.2 concerning the transformation of parallel gateways. Involved actors were the head of chair, the chair's secretariat and the applicant. The two different transformations of the parallel gateway appearing in the process model were the static and the dynamic sequential transformation.

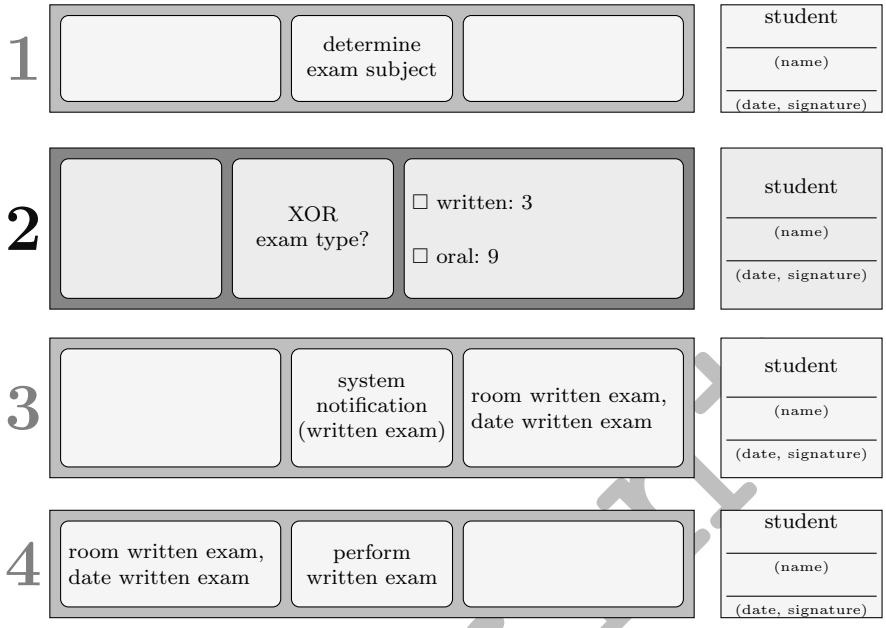

Fig. 13: Parts (first 4 points) of the final graphical checklist for the process "subscribing for an exam" which was used for evaluation.

To do a parallel transformation was not suitable in this context, as the parallel subprocesses were too short to get any substantial advantages of this form except perhaps better values for the parallelism criterion. To get results for the subjective criteria every execution support tool was evaluated by 21 test persons which means a total of 105 evaluations.

6.1 Objective Topics

First of all, the conditions for the three objective topics had to be set up. For the topic of flexibility three possible values are available: A low flexibility value means that during the execution of a process the underlying process model can't be changed. A medium value means that the order of activities can be customized or that certain elements can be deleted. A high flexibility value is assigned, if nearly everything can be adjusted during execution. For example, if the chair's secretary is not accessible this agent can be changed to another person in all activities where necessary for this single process instance. The parallelism values are distributed in the following manner: A low value is allocated if the execution order is determined before process initialization and is therefore only sequential. A medium value means that the execution order is sequential but determinable at run time. Parallelism is high if real parallelism is possible. For the "Subscribing for an exam"-process model no

No.	type	ID/AN	AC/CO	OD/GT	AG
1	o		determine exam subject		student
2	c		XOR exam type?	$s_2 = 0$ $a_{2,1}$ =written $g_{2,1} = 3$ $a_{2,2}$ = oral $g_{2,2} = 9$	student
3	o		system notification (written exam)	{room written exam, date written exam}	student
4	o	{room written exam, date written exam}	perform written exam		student
5	o	{exam unmarked}	perform exam correction	{exam marked}	auditor
6	o	{exam marked}	register exam marks in system		secretariat of chair
7	o	{exam marked}	send exam to examination office		secretariat of chair
8	c		XOR end	$s_8 = 0$ $a_{8,1}$ = go to $g_{8,1} = 15$	student
9	o		system notification (oral exam)		student
10	o		determine and assign examination date	{examination date}	secretariat of chair
11	o	{examination date}	perform oral exam	{minutes of examination (unsigned)}	auditor
12	o	{minutes of examination (unsigned)}	sign minutes of examination	{minutes of examination (signed)}	assessor
13	o	{minutes of examinationl (unsigned)}	sign minutes of examination	{minutes of examination (signed)}	auditor
14	o	{minutes of examination (signed)}	Send exam mark and protocol to examination office		secretariat of chair
15	o		exam notification and performance finished		secretariat of chair

Fig. 14: Checklist vector for the process “Subscribing for an exam” containing all elements according to Definition 2 needed for the graphical checklist.

parallelism values could be distributed as the are no parallel gateways in the BPMN process model.

Length of the process execution support tools is just the number of all flow objects in the BPMN model (activities, events, gateways) or the number of operating and control points in the checklists. This value depends strongly on the underlying process and shall only give an impression of the compactness of the different tools. Only tools with the same underlying process are comparable with each other. Table 1 shows the results of the objective evaluation criteria for the two processes “Subscribing for an exam” and “Applying for a business trip”.

	Subscribe for exam		Apply for business trip		
	BPMN	Checklist	BPMN	ShortChecklist	LongChecklist
Flexibility	low	high	low	high	high
Parallelism	NA	NA	high	low	medium
Length	16	15	20	15	18

Table 1: Values for the objective evaluation criteria for the different support tools; NA = not available.

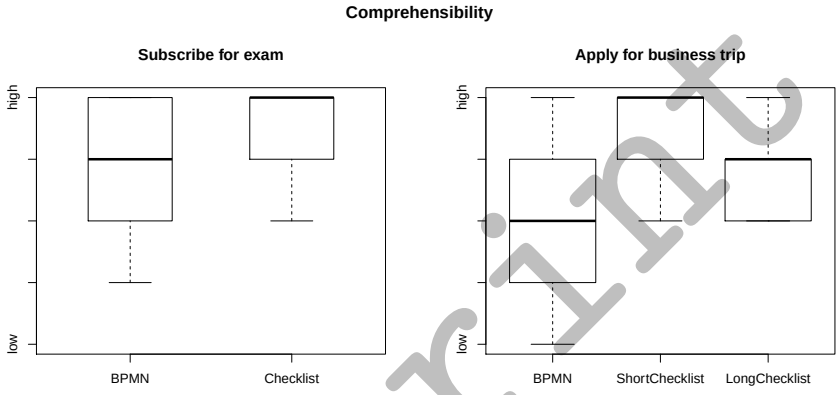

Fig. 15: Boxplot evaluation of the comprehensibility for the two processes "Subscribe for exam" (BPMN model and checklist) and "Apply for business trip" (BPMN model, a short checklist and a long checklist).

6.2 Subjective Topics

As mentioned in the introduction of the current section, the subjective evaluation criteria are comprehensibility, orientation and reliability. The test persons had to go through the two processes with the help of the different process execution tools and afterwards rate their impressions of the three subjective criteria by classifying with a Likert scale of five classes. The generated boxplots[1] allow a good interpretation of the results for the three criteria.

Fig. 15 shows the boxplots for comprehensibility of the different process execution tools for both processes. A high value reflects the opinion of the test person that she completely understands the process and the handling of the execution tool, in contrast to a low value where she neither understands the process nor the tool's handling. As it can be seen in Fig. 15, the answering dispersion for the BPMN models is greater than that of the checklists which could occur from the fact, that some persons easily understand the BPMN notation, but some do not, and that the checklists are understood quite well among all

[1] For more information about boxplots see [4], Section 2.2, and [13].

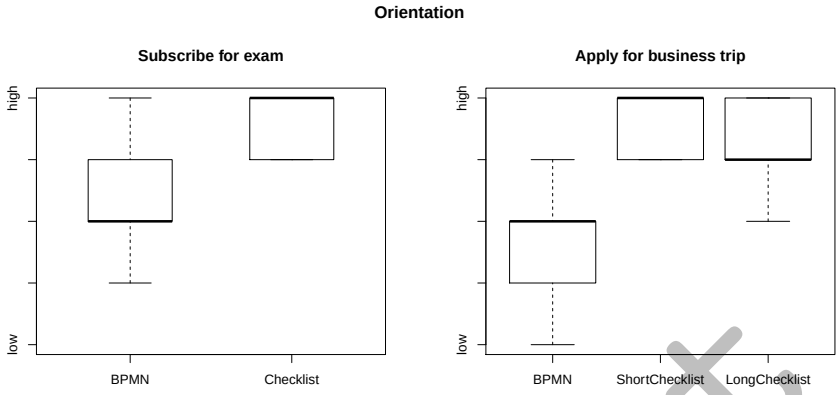

Fig. 16: Boxplot evaluation of the orientation for the two processes "Subscribe for exam" (BPMN model and checklist) and "Apply for business trip" (BPMN model, a short checklist and a long checklist).

test persons. Furthermore, the median of all answers for the checklists is higher than that of the corresponding BPMN-model, although the shorter, but less flexible, checklist for the business trip process seems easier to understand than the longer but more flexible one.

The second subjective evaluation criterion is orientation, which asks for knowing the own position during the execution of the process and the steps that have to be performed next. Again, a set of boxplots for the two processes and the two, respectively three, execution support tools was generated and is shown in Fig. 16. The results for the orientation aspect are similar to that of comprehensibility: Dispersion for the BPMN support tools is higher than for the checklists and values for the median are higher, which means better, for the checklists than for the BPMN execution support tool. As it can be seen, the short checklist for the business trip process has a better orientation value than the long checklist. This could be caused by the fact, the short checklist provides a slightly better overview over the process and clearer instructions for the tasks, as there are fewer control points and thus more sequential task chains which allow for easier orientation. Moreover, one conspicuity is that the difference of the BPMN support tools and the checklists is greater for the orientation aspect than for the first aspect, comprehensibility.

In Fig. 17, boxplots for the third subjective evaluation criterion, reliability, are presented. A high level of reliability means that the course of the process execution so far is traceable and it is clear, who has carried out which tasks. It sticks out that the mean values for the BPMN support tool are quite low and for the checklists have even increased compared to the other evaluation criteria. There is hardly any spread in the responses concerning the checklists which means that they provide a high reliability for nearly all test persons. Responses concerning the BPMN model again vary very much, but at a lower level than

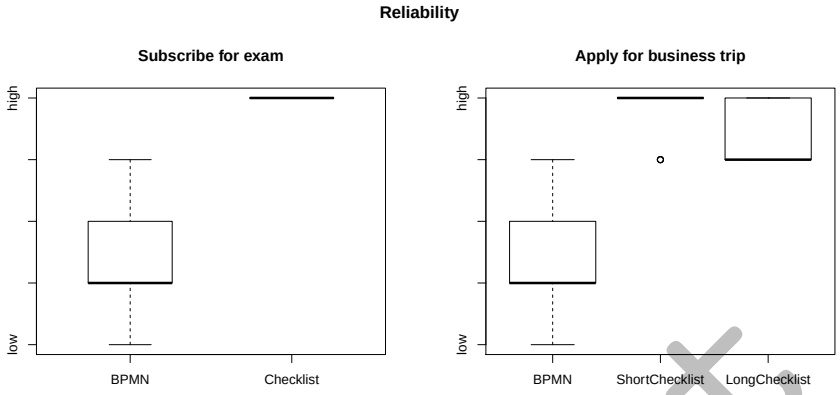

Fig. 17: Boxplot evaluation of the reliability for the two processes "Subscribe for exam" (BPMN model and checklist) and "Apply for business trip" (BPMN model, a short checklist and a long checklist).

before. So, even for the people who understood the BPMN evaluation tool quite well, it does not seem to provide an accordingly good value of reliability during the execution.

7 Conclusion, Limitations and Future Work

In order to diminish the dependency from IT-based process management systems, the work at hand proposed an alternative way of supporting workflow execution that is suitable for human-driven processes. We introduced the process checklist representation of process models where processes are described as a paper-based step-by-step instruction handbook. The process checklist is handed over during process execution from process participant to process participant. Successful task accomplishments are recorded through signatures of corresponding process participants.
This way, the process checklist also supports the key benefits of traditional WfMS. The checklist is handed over to responsible agents (task coordination), process tasks are serialized and marked by a unique identifier (step-by-step guidance) and the checklist itself as well as the corresponding signatures ensure traceable process execution. The work at hand provides the general structure of process checklists as well as a transformation algorithm of basic BPMN process model elements to process checklists. Furthermore, we described implementation details by giving a concrete checklist meta-model as well as a XML-based serialization possibility. The checklist approach has been evaluated in two real-life case studies, i.e., it supported the execution of university processes. The results showed that process checklists serve as a feasible process execution support and are highly accepted by process participants.

In contrast to the advantages over IT-based process management systems as mentioned before, paper-based checklists can also have disadvantages compared to traditional systems. Checklists represent a single point of access, so support for distributed agents may be difficult. If this is the case, one has to ask if using a paper-based checklist is the right thing for this specific application, as we recommend using checklists for example in administrational environments.
In general, it is possible to transform a procedural process model to a process checklist based on the proposed algorithm. However, due to the serialization of the process, the checklist representation has of course problems when dealing with parallelism. Here, process modelers have to choose a suitable transformation method as described in Section 4. The presented case studies focused on a first evaluation in the field of university processes. For future work we will evaluate the proposed approach especially within business cases. Here, we expect useful experiences regarding the acceptance and cooperation of participating agents. Based on these results we will further improve methodology, design and representation. Furthermore, we will focus the transformation of loosely-specified processes like declarative process models, e.g., Declare [12].

Acknowledgements The presented work is developed and used in the project "Kompetenzzentrum für praktisches Prozess- und Qualitätsmanagement", which is funded by "Europäischer Fonds für regionale Entwicklung (EFRE)".
Michael Heinrich Baumann wishs to thank Hanns-Seidel-Stiftung e.V. (HSS).

References

1. M. Baumann, M. H. Baumann, S. Schönig, S. Jablonski: Enhancing Feasibility of Human-driven Processes by Transforming Process Models to Process Checklists. 15th Working Conference of Business Process Modeling, Development, and Support (BPMDS 2014), Springer, 2014
2. C. Condon: The Computer Won't Let Me: Cooperation, Conflict and the Ownership of Information, CSCW, pp. 171-185, 1993
3. J. W. Ely, M. L. Graber, P. Croskerry: Checklists to Reduce Diagnostic Errors, Academic Medicine, 03/2011, Vol. 86, Issue 3, pp. 273-404
4. L. Fahrmeir, R. Künstler, I. Pigeot, G. Tutz: Statistik – Der Weg zur Datenanalyse, Springer-Verlag Berlin Heidelberg, sechste, überarbeitete Auflage, 2007
5. B. M. Hales, P. J. Pronovost: The checklist – a tool for error management and performance improvement, Journal of Critical Care, Vol. 21, Issue 3, pp. 213-235, 2006
6. R. Hauser, M. Friess, J. M. Küster, J. Vanhatalo: Combining Analysis of Unstructured Workflows with Transformation of Structured Workflows, Proc. 10th IEEE International Enterprise Distributed Object Computing Conference (EDOC), 2006
7. S. Jablonski: Do We Really Know How to Support Processes? Considerations and Reconstruction, G. Engels et al. (Eds.): Nagl Festschrift, LNCS 5765, pp. 393-410, 2010, Springer-Verlag Berlin Heidelberg, 2010
8. J. Koehler, R. Hauser, J. Küster, K. Ryndina, J. Vanhatalo, M. Wahler: The Role of Visual Modeling and Model Transformation in Business-driven Development, Proc. 5th International Workshop on Graph Transformation and Visual Modeling Techniques (GT-VMT'06), Vienna, Austria, 2006
9. P. Luff, Ch. Heath, D. Greatbatch: Tasks-in-interaction: paper and screen based documentation in collaborative activity, CSCW, 1992

10. M. Melenovsky: Business process managements success hinges on business-led initiatives, Gartner Research, Stamford, CT, no. July, pp. 1-6, 2005
11. Object Management Group Inc.: Business Process Model and Notation (BPMN) Version 2.0, http://www.omg.org/spec/BPMN/2.0, 2011
12. M. Pešić: Constraint-Based Workflow Management Systems, Shifting Control to Users, 2006
13. http://www.r-project.org/, 2014-09-30
14. M. Reichert, B. Weber: Enabling Flexibility in Process-aware Information Systems, Springer Berlin Heidelberg, 2012
15. W. Van der Aalst, M. Weske, D. Grünbauer: Case handling: a new paradigm for business process support, Data & Knowledge Engineering, 53 (2), pp. 129-162, 2005
16. A. M. Wolff, S. A. Taylor, J. F. McCabe: Using checklists and reminders in clinical pathways to improve hospital inpatient care, MJA 2004, 181, pp. 428-431
17. M. Zairi: Business Process Management: a Boundaryless Approach to Modern Competitiveness. Business Process Management Journal, Vol. 3, pp. 64-80, 1997
18. M. zur Muehlen, J. C. Recker: How Much Language Is Enough? Theoretical and Practical Use of the Business Process Modeling Notation, Z. Bellahsène and M. Léonhard (Eds.): CAiSE 2008, LNCS 5074, pp. 465-479, 2008, Springer-Verlag Berlin Heidelberg, 2008

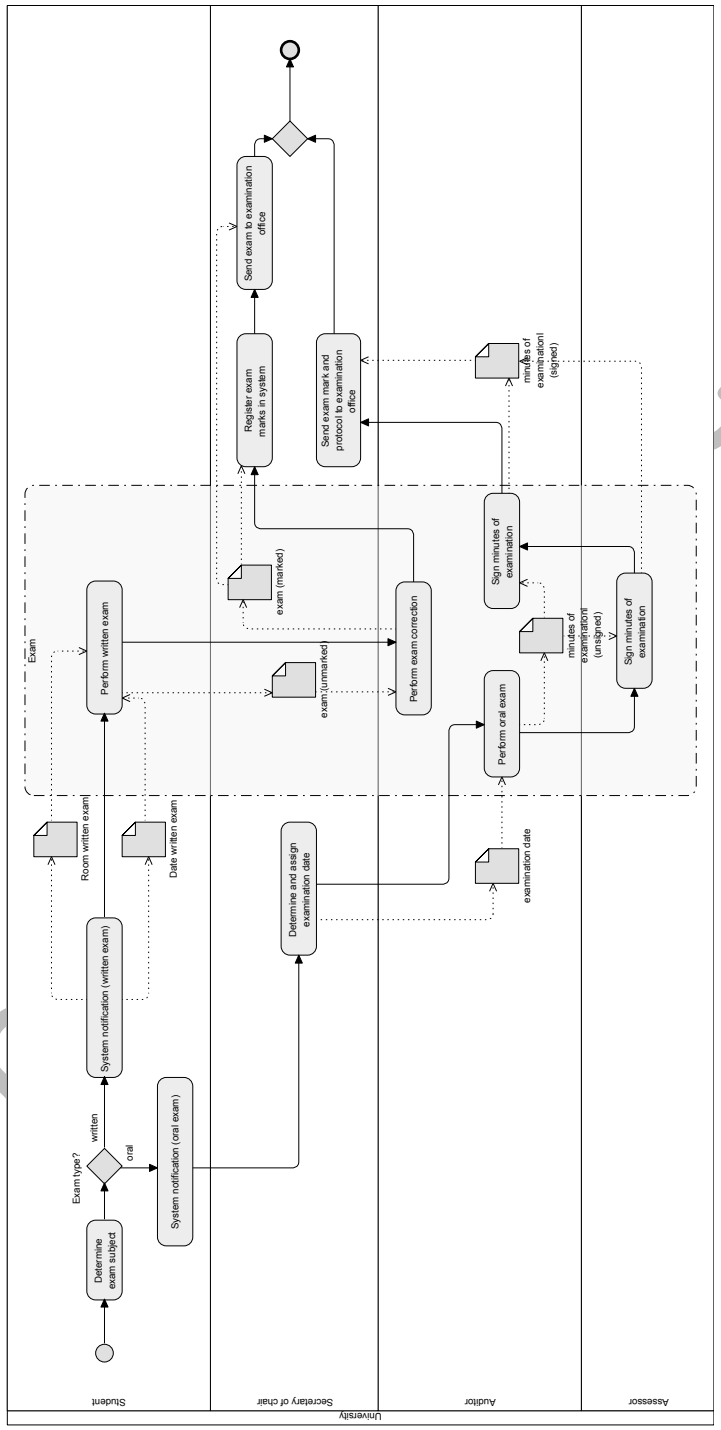

Fig. 18: BPMN model for the process “Subscribing for an exam”.